JavaScript
逆引きハンドブック

古籏 一浩◆著

基本的な処理や便利なTipsとともに、HTML5のAPIを数多く掲載!

まさにWeb制作のバイブル!

HTML5
CSS3
対応!

「やりたいこと」からJavaScriptの機能を探せる!

24時間無料でサンプルデータをダウンロードできます。

C&R研究所

JavaScript 逆引きハンドブック

古籏 一浩◆著

HTML5
CSS3
対応!

C&R研究所

■権利について

- 本書に記述されている社名・製品名などは、一般に各社の商標または登録商標です。
- 本書では™、©、®は割愛しています。

■本書の内容について

- 本書は著者・編集者が実際に操作した結果を慎重に検討し、著述・編集しています。ただし、本書の記述内容に関わる運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- 本書で紹介しているコードの実行結果の画面などは、環境によって異なる場合がございますので、あらかじめ、ご了承ください。
- 本書の内容は、2012年7月現在の情報を基に記述しています。仕様の変更やブラウザのバージョンアップなどにより、サンプルの動作が変わったり、プログラムの書き換えが必要になったり、動作しなくなったりする場合があります。また、紹介しているURLなども変更になる場合があります。あらからじめ、ご了承ください。

■サンプルについて

- 本書で紹介しているサンプルは、C&R研究所のホームページ(http://www.c-r.com)からダウンロードすることができます。ダウンロード方法については、5ページを参照してください。
- サンプルデータの動作などについては、著者・編集者が慎重に確認しております。ただし、サンプルデータの運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- サンプルデータの著作権は、著者及びC&R研究所が所有します。許可なく配布・販売することは堅く禁止します。

●本書の内容についてのお問い合わせについて

この度はC&R研究所の書籍をお買いあげいただきましてありがとうございます。本書の内容に関するお問い合わせは、FAXまたは郵送で「書名」「該当するページ番号」「返信先」を必ず明記の上、次の宛先までお送りください。お電話や電子メール、または本書の内容とは直接的に関係のない事柄に関するご質問にはお答えできませんので、あらかじめご了承ください。

〒950-3122 新潟県新潟市北区西名目所4083-6　株式会社 C&R研究所　編集部
FAX 025-258-2801
「JavaScript逆引きハンドブック」サポート係

PROLOGUE

今日、JavaScriptは、Web制作において欠かせないものとなりました。今やJavaScriptが使われていないWebサイトを探す方が大変でしょう。

当初はフォームの入力チェック程度にしか使われていなかったJavaScriptが、現在ではWebアプリケーションまで担うようになったのはすごいことです。一時期、JavaScriptを動作しないようにブラウザを設定するような時代もありました。また、ブラウザ間の動作の違いによる問題に悩まされた時代もありました。今では、jQueryなど各種ライブラリの利用によってかなり解消されています。

そして、JavaScriptが誕生してから16年、さまざまな困難を乗り越えて現在に至っています。

特にその原動力となったのはAjax(非同期通信)、そしてHTML5によるところが大きいでしょう。HTML5のAPIを利用することにより、高度で複雑なことが簡単にできるようになりました。数年前は不可能だったことがJavaScriptで簡単にできるようになっているのです。たとえば、次のような機能をJavaScriptから使うことができます。

- 位置情報
- 2D画像処理、3D描画
- ビデオ再生/オーディオ再生
- 音声/波形処理
- カメラ撮影/ビデオ撮影/録音
- カメラからのリアルタイムキャプチャー
- 並列処理
- ファイル処理
- バッテリーやセンサーなどデバイス関係
- ソケット通信(双方向通信)
- データベース(Key Value Storeタイプ)

HTML5のAPIではさまざまな機能や仕様が追加・定義され、多くのブラウザで利用できるようになっています。特にスマートフォンで利用できるというのは、大きなポイントです。

本書は、最新のHTML5のAPIを利用したサンプルを数多く掲載しています。また、JavaScriptの基本的な処理や便利なTIPSなども多数、掲載しています。

HTML5は進歩が早い上に仕様が確定していないこともあり、将来的に動作しない可能性もあります。それでも、現時点で動作するという点において役立つ部分もあるはずだと考え、掲載しています。

本書が読者の皆様のWeb制作の一助となれば幸いです。

2012年7月

古籏一浩

本書について

動作確認の環境について

本書では、次のような環境で動作を確認しています。

- IE6～8：Windows XP
- IE9：Windows 7/Windows Phone 7.5
- IE10：Windows 8(Release Preview)
- Google Chrome 21：Windows7/Mac OS X 10.6/Mac OS X 10.7
- Chrome for Android：Android 4.0
- Firefox 14：Windows7/Mac OS X 10.6/Mac OS X 10.7
- Firefox(Android版)：Android 4.0
- Safari 5.x：Mac OS X 10.6/Mac OS X 10.7
- Safari 6：Mac OS X 10.8
- Safari(iOS版)：iOS 4/iOS 5.1
- Android 標準ブラウザ：Android2.3/3.2/4.0
- Opera 11～12：Windows 7

スマートフォン・タブレットについては、次の機種で確認しています。

- iOS 4/5：iPhone 4/4S、iPad 2/New iPad
- Android 2.3：Galaxy S/S2、AQUOS PHONE(SH12C)、XPERIA(SO-01C)
- Android 3.2：SONY Tablet(SGPT111JP/S)、Arrows Tab(F-01D)
- Android 4.0：Galaxy Nexus
- Windows Phone 7.5：Windows Phone IS12T

本書の表記方法

本書の表記についての注意点は、次のようになります。

▶サンプルコードの中の「\」(バックスラッシュ)と「¥」(円記号)について

環境によっては、「\」(バックスラッシュ)は「¥」(円記号)で表示されます。本書の誌面に掲載しているサンプルコードでは「\」で表記していますので、必要に応じて読み替えてください。なお、Unicodeでは「¥」と「\」は異なる文字コードになっています。このため、一部の書体を除き、「\」を「¥」で入力すると動作しません。

▶サンプルコードの中の▼について

本書に記載したサンプルコードは、誌面の都合上、1つのサンプルコードがページをまたがって記載されていることがあります。その場合は▼の記号で、1つのコードであることを表しています。

▶ ブラウザの対応バージョンについて

本書では項目のタイトル部分に、ブラウザの対応バージョンのマークを記載しています。「WP」はWindows Phoneを表しています。基本的には記載しているバージョン以降でその項目(メソッドなど)が対応となりますが、一部、例外があります。その場合は、HINTやCOLUMNに記載しています。なお、薄く表示されている場合は、非対応となります。



Ⅲ サンプルファイルのダウンロードについて

本書のサンプルデータは、C&R研究所のホームページからダウンロードすることができます。本書のサンプルを入手するには、次のように操作します。

❶「http://www.c-r.com/」にアクセスします。

❷トップページ左上の「商品検索」欄に「108-5」と入力し、[検索]ボタンをクリックします。

❸検索結果が表示されるので、本書の書名のリンクをクリックします。

❹書籍詳細ページが表示されるので、[サンプルデータダウンロード]ボタンをクリックします。

❺下記の「ユーザー名」と「パスワード」を入力し、ダウンロードページにアクセスします。

❻「サンプルデータ」のリンク先のファイルをダウンロードし、保存します。

サンプルのダウンロードに必要なユーザー名とパスワード

ユーザー名	**jsgh**
パスワード	**f6h3t**

※ユーザー名・パスワードは、半角英数字で入力してください。また、「J」と「j」や「K」と「k」などの大文字と小文字の違いもありますので、よく確認して入力してください。

Ⅲ サンプルコードの利用方法

サンプルファイルは、CHAPTERごとのフォルダの中に、項目番号のフォルダに分かれています。サンプルはZIP形式で圧縮してありますので、解凍してお使いください。

それぞれのフォルダ内には、HTMLファイルやJavaScriptファイル、CSSファイルなどが保存されています。

なお、サンプルによっては、ローカルディスク上(HTMLファイルを直接、ブラウザで開く方法)では動作しない場合があります。その場合は、動作の確認を行うにはネットワーク環境(Webサーバー上)で確認してください。Webサーバーについては、ローカルサーバーでも構いません。

また、本書のサンプルでは、主に「addEventListener()」メソッドを使っています。IE6～8では「addEventListener()」メソッドがないため、そのままでは動作しません。IE6～8では、「attachEvent()」メソッドを使うように書き換える必要があります(400ページ参照)。

CONTENTS

CHAPTER 04 数値・数学

CONTENTS

CHAPTER 05 文字列・正規表現

CHAPTER 06 日付・時刻

CHAPTER 07 ブラウザオブジェクト

CHAPTER 10 Canvas/Canvas 3D/SVG

CHAPTER 12 File API

CHAPTER 13 HTML5のその他のAPI

CHAPTER 01

JavaScriptの基礎知識と基本文法

JavaScriptの概要

JavaScriptとは

JavaScriptは数あるスクリプト言語の1つです。JavaScriptは多くのブラウザに搭載されており、Webアプリケーションの実現に欠かせない言語となっています。特にHTML5の登場で3D処理や並列処理、カメラからのリアルタイム取り込みなど、これまでよりも大幅に実現できる処理が増えており、従来のJavaScriptとは役割も作成されるプログラムの規模や複雑さも大きく変わってきています。このため、作成したJavaScriptプログラムが正しく動作するかをテストするJSUnit(http://www.jsunit.net/)やQUnit(http://qunitjs.com/)などのツールも登場しています。もちろん画像を入れ替えるだけ、入力フォームのチェックを行うだけといった従来通りの簡単な処理もできます。

▶さまざまな場面で使われている

JavaScriptはInternet ExplorerやFirefox、Safari、Google Chromeなどのブラウザに搭載されているため、ブラウザ専用のスクリプト言語と思われがちです。しかし、JavaScriptはOSであるWindows、Adobe Photoshopなどのアプリケーション、Adobe FlashやPDFなどにも搭載されており、さまざまな場面で使われています。また、HTML5(http://www.w3.org/TR/html5/)の普及に伴ってスマートフォン上で動作するアプリケーションを駆動する言語としても利用され始めています。たとえば、Titanium(http://www.appcelerator.com/)やPhoneGap(http://phonegap.com/)などのスマートフォン向けアプリ開発のフレームワークでJavaScriptが使われています。

▶JavaScriptの基本仕様

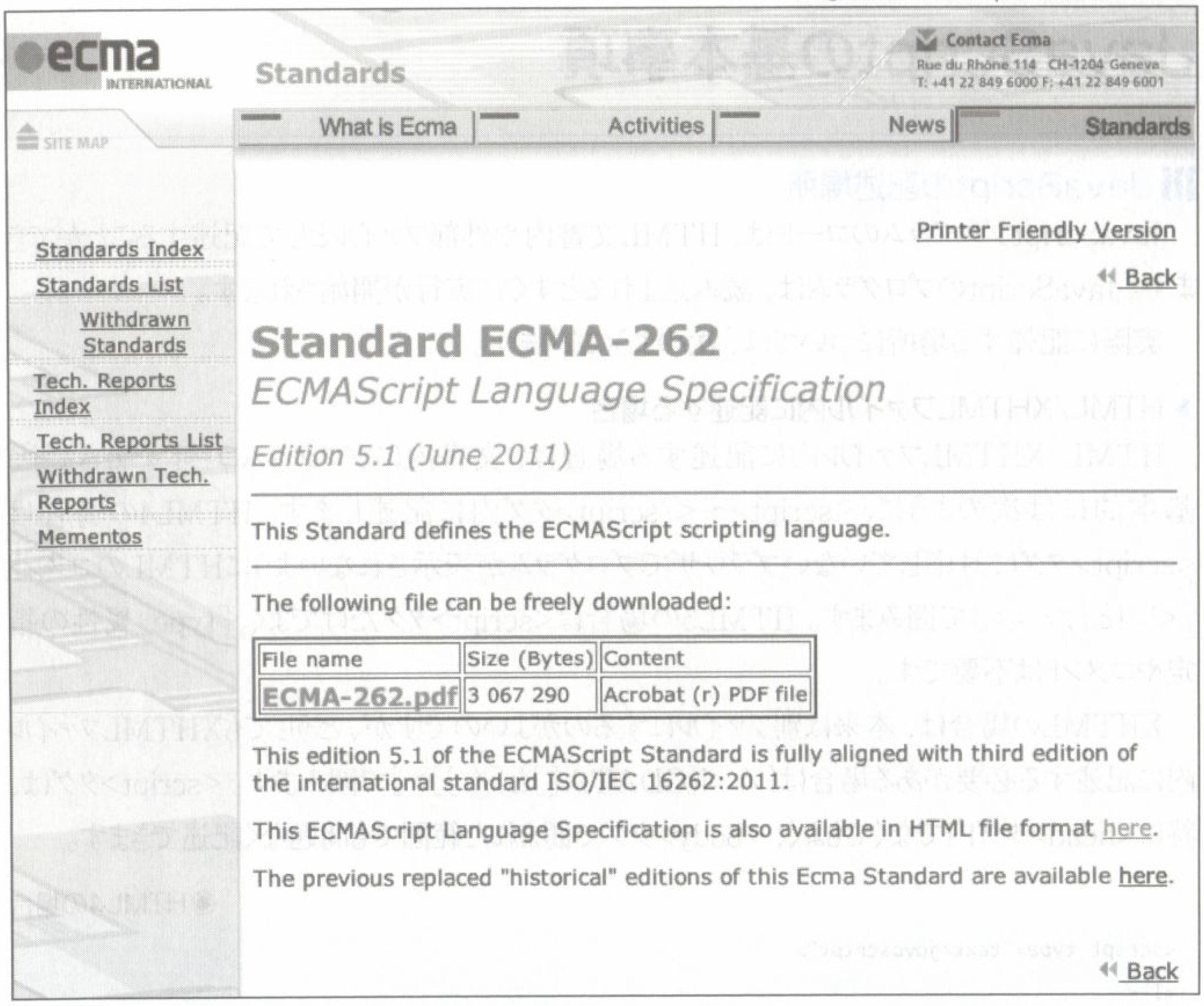
JavaScriptの基本となる部分の仕様はECMAにより公開されており、ECMA Scriptと呼ばれています。この基本部分にブラウザやアプリケーションが独自のオブジェクトを追加したものが、一般的にいわれるJavaScriptです。

また、マイクロソフト社の場合は、JavaScriptと互換性を持った独自拡張のスクリプト言語であるJScriptとしてブラウザなどに実装されています。JScriptではWindows固有のActive Xを利用した処理(ファイル処理などローカル環境の制御など)も可能になっています。ただし、HTML5の普及に伴って直接、ファイル処理やデバイスへのアクセスが可能になってきているため、Active Xを利用した処理の必要性は低下しています。

- ECMA Script-262

URL http://www.ecma-international.org/publications/standards/Ecma-262.htm

◉ECMA Script-262のページ



▶プロトタイプ指向のオブジェクト指向言語

JavaScriptはJavaなどの一般的なクラスを利用したものと異なり、基本となるオブジェクトを拡張・派生させるプロトタイプ指向のオブジェクト指向言語です。ただし、当初、実装されたブラウザであるNetscape 2と現時点のものでは異なっている部分が多くあります。当初はプロトタイプによる継承もなく、単純なオブジェクト指向言語で、変数名や文字数制限、利用できるオブジェクトなど、多くの制限がありました。これが、次第に改良されて現在に至っています。

▶ライブラリ

ブラウザの普及とともにJavaScriptも改良されたとはいえ、ブラウザによって動作が異なることも多く、制作者泣かせの言語であることも確かです。これを解消するのがJavaScriptライブラリです。2012年現在で最も多く利用されているのがjQueryライブラリです。jQueryは短いコードで多くの処理を実現できるのが魅力の1つです。また、UI処理を行うためのライブラリやスマートフォン向けのjQuery Mobileも用意されています。その他のJavaScriptライブラリとしては中規模・イントラネット向けのExtJS(SenchaTouch)、大規模向けのDojo ToolKitなどがあります。

JavaScriptの基本事項

JavaScriptの記述場所

JavaScriptプログラムのコードは、HTML文書内や外部ファイルとして記述することができます。JavaScriptのプログラムは、読み込まれるとすぐに実行が開始されます。

実際に記述する場所については、次のようになります。

▶HTML/XHTMLファイル内に記述する場合

HTML/XHTMLファイル内に記述する場合は、文書内にプログラムコードを書きます。基本的には次のように、<script>〜</script>タグ内に記述します。HTML4の場合は<script>タグに対応していないブラウザでプログラムが表示されないようにHTMLのコメント「<!--」と「// -->」で囲みます。HTML5の場合は<script>タグだけでよく、「type」属性の指定やコメントは不要です。

XHTMLの場合は、本来は別ファイルにするのがよいのですが、どうしてもXHTMLファイル内に記述する必要がある場合は「//<![CDATA[」と「//]]>」で囲みます。<script>タグは、特に<head>タグ内でなくてもよく、<body>タグで囲まれた範囲でも問題なく記述できます。

◉HTML4の場合

```
<script type="text/javascript">
<!--
  ここにスクリプトを書く
//-->
</script>
```

◉HTML5の場合

```
<script>
  ここにスクリプトを書く
</script>
```

◉XHTMLの場合

```
<script type="text/javascript">
//<![CDATA[
  ここにスクリプトを書く
//]]>
</script>
```

また、HTML4でデフォルトのスクリプト言語としてJavaScriptを指定する場合は、次の<meta>タグを指定しておきます。

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

▶ 外部ファイルとして記述する場合

外部ファイルとして使用する場合は、純粋にスクリプトだけを記述したテキストファイルを作成し、<script>タグの「src」属性でファイル名を指定します。これはHTML4でもHTML5でも同様です。

```
<script src="ファイル名.js"></script>
```

ファイル名の末尾(拡張子)は「.js」にする必要があります。「src」属性には拡張子が「.js」のファイルだけでなく、CGIのパスなどを記述して、結果をJavaScriptコードで返すこともできます。また、JSON形式など、データだけを取得したい場合にも利用することができます。一度、読み込まれたスクリプトはメモリ内に残るため、<script>タグを削除しても実行することが可能です。

また、外部ファイルの場合は「同期」で読み込まれるため、プログラムサイズが巨大であったり、解析に時間がかかったりすると、ブラウザの反応速度が低下し、ページ表示までに時間がかかる場合があります。そこで、<script>タグには、非同期でスクリプトを読み込ませて実行する属性が用意されています。ファイルを非同期で読み込ませる場合には、<script>タグに「async」属性(非同期で読み込まれ実行)や「defer」属性(非同期で読み込み、実行はページ構築後)を指定します。ただし、「async」属性を指定すると、実行順序が保証されないので注意が必要です。

HTML5のWeb Workersを利用している場合は、「importScripts()」メソッドにより、スクリプトファイルを読み込ませることができます。この場合、Web Workersが非同期なので、メイン処理とは別に読み込まれます。ただし、「importScripts()」メソッドで読み込むスクリプトの読み込み順序と実行順序は保証されます。

▶ タグ内に記述する場合

タグ内に記述する場合には、イベント名を記述し、「=」の後ろに実行するプログラムを指定します。イベント名は「on」を付加した名称になります。たとえば、「click」イベントをタグ内に記述する場合は、「onclick」となります。多くの場合、関数を呼び出すようになっていますが、通常のプログラムコードをそのまま記述することができます。

```
<a href="#" onclick="alert('ok?')">〜</a>
```

「onclick」や「onmouseover」などのイベント名は、大文字で書いても小文字で書いても動作します。大文字と小文字が混在していても問題ありません。ただし、プログラムでイベントハンドラとして定義する場合にはすべて小文字にする必要があります。

演算子について

JavaScriptで使える演算子

JavaScriptには基本的な演算子のみ用意されており、高度な演算を行う場合は「Math」オブジェクトを利用するようになっています。JavaScriptで使用できる演算子は、次の表のようになります。ビットシフトなどのビット演算は32ビット長で処理されます。ビット演算以外の計算に関してはIEEE 754の規格に沿って処理が行われます。IEEE 754の規格については、次のURLを参照してください。

- IEEE 754 - Wikipedia

URL http://ja.wikipedia.org/wiki/IEEE_754

演算子	意味	記述例	演算の結果（x=12、y=4とした場合）
+	加算	z = x + y	16
−	減算	z = x − y	8
*	乗算	z = x * y	48
/	除算	z = x / y	3
%	剰余	z = x % y	0
++	プリインクリメント	y = ++x	y=13、x=13（代入前に加算）
++	ポストインクリメント	y = x++	y=12、x=13（代入後に加算）
−−	プリデクリメント	y = −−x	y=11、x=11（代入前に減算）
−−	ポストデクリメント	y = x−−	y=12、x=11（代入後に減算）
−	符号反転	y = −x	−12
&	論理積	15 & 9	9（1111 & 1001 = 1001）
&&	論理積（論理式）	(x == 12) && (y == 4)	true
|	論理和	15 | 9	15（1111 | 1001 = 1111）
||	論理和（論理式）	(x == 12) || (x == 9)	true
^	排他的論理和	15 ^ 9	6（1111 ^ 1001 = 0110）
!	否定	!(x == 12)	false
<<	符号付き左シフト	9<<2	36（1001<< = 100100）
>>	符号付き右シフト	9>>2	2（1001>> = 10）
>>>	符号なし右シフト	19>>>2	4（10011>>> = 100）

代入演算子

JavaScriptには、代入演算子も用意されています。使用できる代入演算子は、次の表のようになります。

代入演算子	記述例	「=」を使った記述
=	x = y	なし
+=	x += y	x = x + y
−=	x −= y	x = x − y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y

比較演算子

比較演算子は、値の比較を行います。結果は「true」か「false」となります。

比較演算子	意味
==	式が等しい
===	式が等しく、型も等しい
!=	式が等しくない
>	左が大きい
>=	左が等しいか大きい
<	右が大きい
<=	右が等しいか大きい

三項演算子

JavaScriptには、条件によってどちらか1つの値を取る三項演算子も用意されています。三項演算子は、「?」と「:」(コロン)で区切って指定します。たとえば、変数「b」の値に応じて代入する文字列を変えるには、次のように記述します。

```
a = (b > 2) ? "2以上" : "2未満"
```

条件を満たした場合には「?」直後の値が代入され、条件を満たしていない場合には「:」より後ろの値が代入されます。なお、三項演算子については、59ページも参照してください。

演算子の優先順位

演算子には優先順位が決められています。優先順位は、次の表のようになります。なお、表の上が優先順位が高く、下になるほど優先順位が低くなります。

優先順位	演算子/特殊演算子	記号/キーワード
1	メンバー、オブジェクト生成	. [ ] new
2	カッコ/関数呼び出し	( )
3	インクリメント、デクリメント	++ --
4	否定(NOT)、正数記号(+)、符号反転(−)	! ~ + − typeof void delete
5	乗算、除算、剰余	* / %
6	加算、減算	+ −
7	ビットシフト	<< >> >>>
8	大小、プロパティチェック	< <= > >= in instanceof
9	比較	== != === !==
10	ビット演算/論理積(AND)	&
11	ビット演算/排他的論理和(XOR/EOR)	^
12	ビット演算/論理和(OR)	|
13	論理的AND	&&
14	論理的OR	||
15	条件(三項演算子)	?:
16	代入	= += −= *= /= %= <<= >>= >>>= &= ^= |=
17	カンマ	,

COLUMN　予約語について

JavaScript(ECMA Script ver 5)では、次のキーワードは予約語となっているため、変数名などで使用しないようにしてください。なお、constやletなど、一部はすでにFirefoxなどのブラウザで使用されています。

- class
- const
- enum
- export
- extends
- implements
- import
- interface
- let
- package
- private
- protected
- public
- static
- super
- yield

リテラルについて

JavaScriptのリテラル

リテラルは定数という意味で、その名通り、決まっている値です。プログラム中に直接、記述可能な文字列や数値などです。JavaScriptのリテラルは、次のようになります。

▶数値

数値は8進数、10進数、16進数や、eを使った指数表記などを使用することができます。ECMA-262 5.1thでは、10進数と16進数の数値となります。Strictモードの場合も、10進数と16進数です(160ページ参照)。

▶文字列

文字列は、「'」(シングルクォーテーション)、または、「"」(ダブルクォーテーション)で囲んで表現します(194ページ参照)。

▶真偽値

「真」か「偽」を表します。値は「true」(真)、または、「false」(偽)となります(116ページ参照)。

▶正規表現

正規表現は、「/」(スラッシュ)で囲んで表現します(213ページ参照)。

▶配列

配列は、「[」と「]」の間に、値を「,」(カンマ)で区切って記述します(120ページ参照)。

▶オブジェクトリテラル

オブジェクトは、「{」「}」の間にプロパティと値をセットにして「key : value」の形式で、「,」(カンマ)で区切って記述します(90ページ参照)。

▶関数リテラル(無名関数/匿名関数)

関数についても、関数リテラルを使用することができます。これは、無名関数や匿名関数と呼ばれています(62ページ参照)。

▶特殊記号やUnicode文字

改行コードやタブなどの特殊記号やUnicode文字は、「\値」の形式で記述します。

▶null

「null」(ヌル)は、何もないことを示します。

strictモードとバージョンの指定について

JavaScriptの実行バージョンの指定について

JavaScriptには、いくつかのバージョンがあります。Firefoxでは、「script」要素の「type」属性で明示的にバージョンを指定することができます。たとえば、次のようにするとJavaScriptのバージョン1.7を使用することを指定できます。

```
<script type="text/javascript;version=1.7">
```

「1.7」の部分がバージョンになるので、他のバージョンを指定する場合は、この値を変えればよいことになります。JavaScriptのバージョン指定はFirefoxのみ対応しているため、他のブラウザでは利用できません。

strictモードについて

IE6～9を除くブラウザでは、JavaScriptの実行モードを指定することができます。これはstrictモードと呼ばれ、JavaScriptの曖昧さを回避できるため、プログラム作成上のミスを減らすことができます。strictモードを使用するには、「"use strict";」とだけ記述します。関数内だけ、strictモードを有効にすることもできます。「"use strict";」を記述すると、次のような制限や違いが発生します。

- varなしでグローバル変数に代入するとエラー
- 8進数表記の禁止
- 「with」の使用不可
- 「arguments.callee」の使用不可
- 「eval()」スコープ範囲の違い
- 「NaN」などに代入するとエラー
- 「eval()」、「arguments」の名前に対してバインドや代入不可
- 関数での「arguments」はエイリアスにならずに当初の引数を保持
- 関数内での「this」の扱いの変更(指定された「this」を変更しない)
- 書き込み不可のプロパティに代入するとエラー
- 拡張不可のオブジェクトへの新規プロパティ作成でエラー
- 削除できないプロパティを削除するとエラー
- 重複するオブジェクトリテラルがあるとエラー
- 関数/メソッドのパラメーター名が重複するとエラー
- 関数内で「関数名.caller」「関数名.arguments」指定はエラー
- 「implements」「interface」「let」「package」「private」「protected」「public」「static」「yield」の単語は予約語のため使用不可

なお、iOS 5、Androidでもstrictモードに対応しています。ただし、エラーとなった場合、「try...catch」文で捕捉できないことがあります(次のサンプルも動作結果が異なる)。

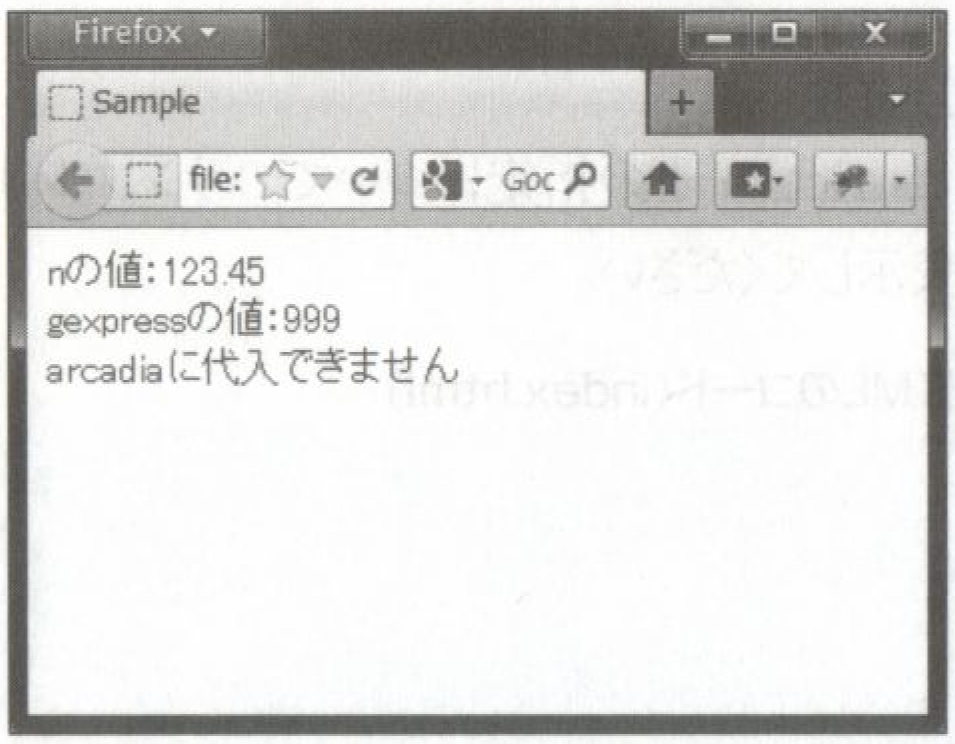
Ⅲ サンプルでの動作の確認

実行バージョンの指定とstrictモードの指定についてのサンプルは、次のようになります。各ブラウザでindex.htmlを表示してください。

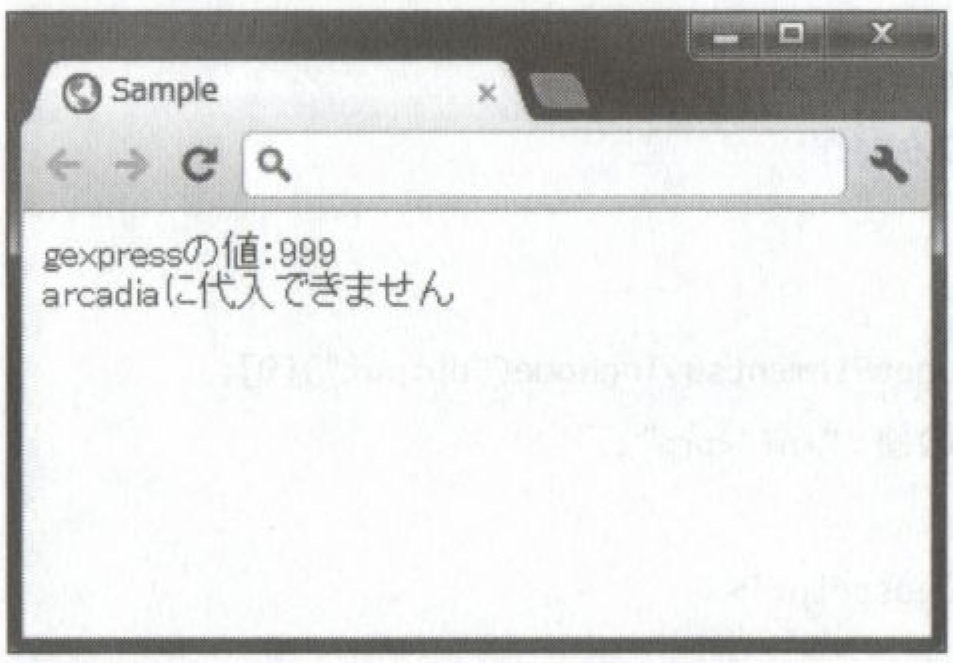
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
  </head>
  <body>
    <output></output>
    <script type="text/javascript;version=1.7">
      // letはFirefoxでのみ動作
      let n = 123.45;
      // nの内容を出力
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML = "nの値 :"+n+"<br>";
    </script>
    <script type="text/javascript">
      // "use strict";
      var ele = document.getElementsByTagName("output")[0];
      // ここはstrictモードではなく通常動作。グローバル変数に代入可能
      gexpress = 999;
      ele.innerHTML += "gexpressの値 :"+gexpress+"<br>";
      // 以下の関数はstrictモード
      function execTest(){
        // strictモードに設定
        "use strict";
        try{
          // strictモードではvarなしグローバル変数代入はエラー
          arcadia = 5;
          ele.innerHTML += "arcadiaの値 :"+arcadia+"<br>";
        }catch(e){
          ele.innerHTML += "arcadiaに代入できません<br>";
        }
      }
      execTest();
    </script>
  </body>
</html>
```

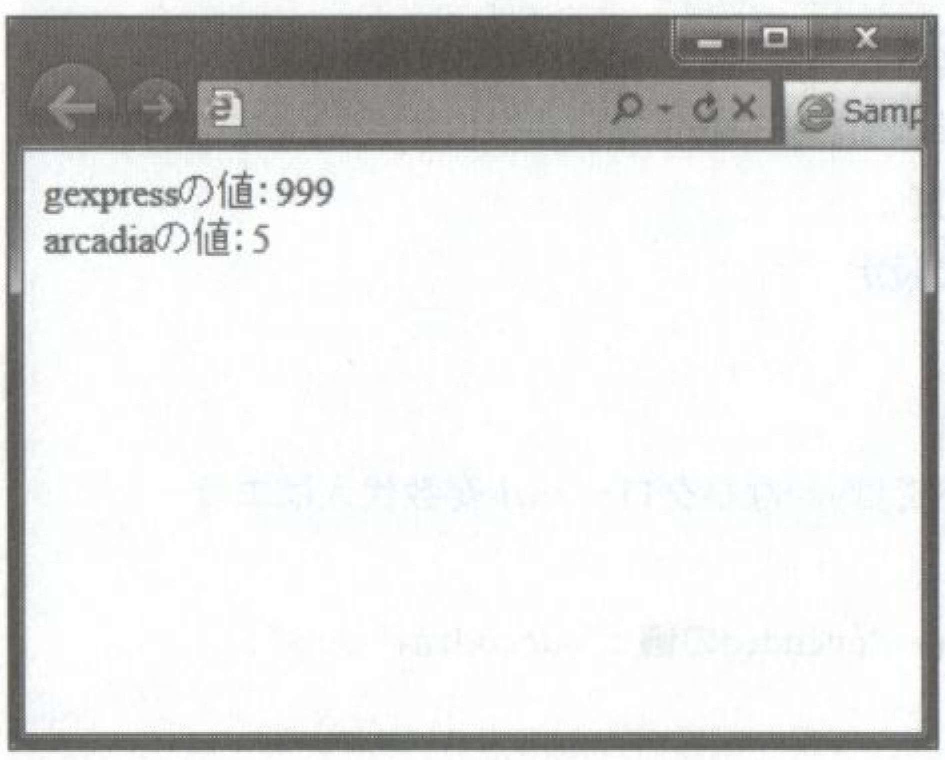
Firefoxでは JavaScriptのバージョンを指定して処理することができるため、変数「n」の値が表示されています。



Firefox、Safari、Opera、IE10、Google Chromeではstrictモードに対応しているため、varなしでグローバル変数に代入するとエラーになります。



IE9ではstrictモードに対応していないため、グローバル変数に代入した結果が表示されます。



コメントについて

JavaScriptのコメント

JavaScriptのコメント(注釈)は、2種類が用意されています。1つは1行コメントである「//」です。これは、「//」より後ろに続く文字から行末(改行コード)までをコメントとするものです。もう1つが複数行をコメントとする「/*~*/」です。「/*」と「*/」で囲まれた部分はすべてコメントとみなされ、処理されません。

```
var a; // これ以降は行末(改行コード)までをコメントとする

var b; /* この部分はコメントとなる */

/*
  このコメントでは、複数行のコメントを
  記述することができる

*/
```

たとえば、次のコードでは最初のアラートダイアログはコメントになっていないので表示されますが、その次の行にある「alert()」メソッドはコメントになっているので表示されません。また、ページ内にOKの文字が表示されますが、以後に続く処理はコメントになっているので処理されません。

SAMPLE CODE HTMLのコード(index.html)

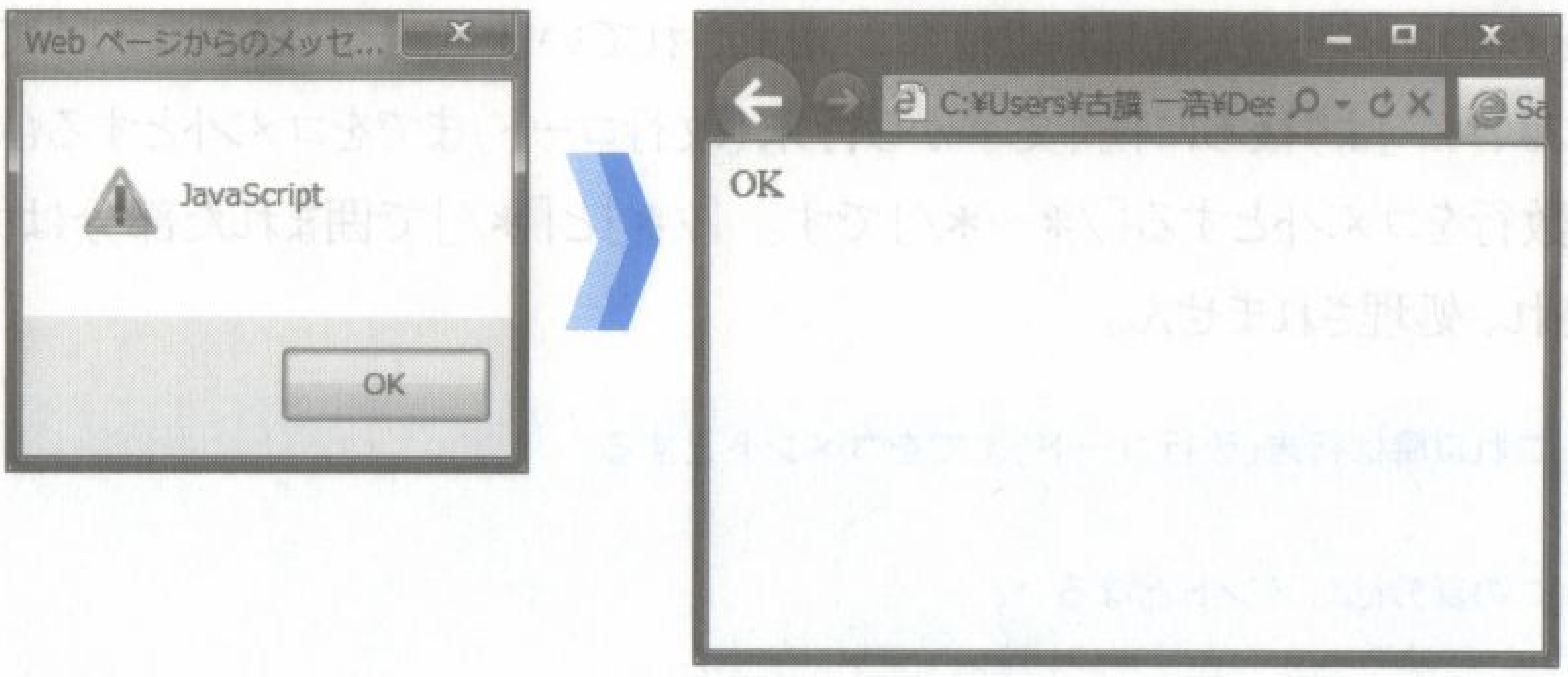
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  alert("JavaScript");
  // alert("OK");
  document.getElementsByTagName("output")[0].innerHTML = "OK";
  /*
  document.getElementsByTagName("output")[0].innerHTML = "NG";
```

```
    alert("NG!");
    */
}, false);
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



変数と定数について

JavaScriptでの変数の定義

JavaScriptでの変数宣言には、いくつかの方法があります。JavaScriptでは、「var」や「const」を使わずに、いきなり変数を使用することができます。「var」宣言しない場合、変数はグローバル変数として定義されます。また、関数外では「var」宣言してもしなくても、グローバル変数として定義されます。グローバル変数ではなく、ローカル変数として宣言する場合には、関数ブロック内で「var」を使って宣言します。ブロック内でのみ有効な変数にしたい場合には「let」がありますが、これはFirefoxでしか使用することができません。

変数をまとめて宣言する場合は、「,」(カンマ)で区切って1行で記述します。また、変数の宣言と同時に初期値を代入することもできます。これを変数の初期化といいます。

```
// 変数「apple」を宣言する
var apple;

// 変数「banana」「cherry」をまとめて宣言する
var banana, cherry;

// 宣言と同時に初期値を指定する
var apple = 123;

// 複数の変数にそれぞれ初期値を指定してまとめて宣言する
var banana = 456, cherry = 789;

// var宣言なしで変数「orange」を定義する(グローバル変数として定義される)
orange = "OK";

// ブロック内でのみ有効な変数を定義する(Firefoxのみ)
let n ="NG";
```

▶ 変数名の制限

変数名については、次の制限があります。

- 最初が英文字、または、「_」(アンダーバー)で始まる
- 2文字目以降は、英数字と「_」(アンダーバー)を使用することができる
- 大文字と小文字は区別される(例：「abc」と「Abc」は別の変数とみなされる)
- 予約語を変数名にすることはできない

```
// 次の変数は有効
var abc // 英字で始まる
var _abc // 「_」から始まる
var abc_123;
```

▼

```
// 変数「abc」と変数「Abc」は区別される
var abc;
var Abc;

// 次の変数は無効
var 123; // 数字で始まる
var var; // 予約語
```

▶代入

変数への値の代入は、「=」演算子を使います。また、「a = b = c = 123;」のように一括して同じ内容を代入することもできます。

```
a = 123;            // 変数「a」に「123」を代入する
a = b;              // 変数「a」に変数「b」の値を代入する
a = b = c = 123;    // 変数「a」「b」「c」に「123」を代入する
```

▶変数で扱えるデータ

JavaScriptの変数は非常に柔軟な仕様になっており、どんなデータ(数値や文字列、関数、オブジェクト、メソッド、プロパティ)でも入れることができます。つまり、変数の型を問いません。たとえば、「a = 12.3」と変数「a」に数値の「12.3」を入れた後で、「a = "OK"」としても問題ありません。ただし、変数の型を問わないのは非常に便利な反面、型の不一致によるエラーや思わぬ結果を引き起こすことがあります。これを防ぐには、変数の型を調べてから処理するのが安全です。変数の型を調べる方法については、112ページを参照してください。

▶注意点

変数に数値や文字列を代入した場合はそのものが変数になりますが、オブジェクトなどの場合は注意が必要です。変数にオブジェクトを代入すると、オブジェクトそのものがコピーされるのではなく、オブジェクトへの参照が代入されます。

たとえば、次のコードでは、変数「a」にページ全体のスタイルシートオブジェクトへの参照を代入しています。その後、変数「b」に変数「a」を代入していますが、ページ全体のスタイルシートオブジェクトが複製されて変数「b」に代入されているわけではありません。単純にオブジェクトへの参照だけが変数「b」に代入されます。このため、コードを実行すると、ページ全体の背景色が赤になります。「b = a」は「b = document.body.style」と等価になるということです。

```
a = document.body.style;
b = a;
b.backgroundColor = "red";
```

また、ブラウザに実装されているJavaScriptでは、単純に変数を作成すると、「window」オブジェクトのプロパティとして追加されます。つまり、グローバル変数という名目になっていますが、実際には「window」オブジェクトのプロパティとして割り当てられています。これは、「for...in」文で「window」オブジェクトの中身を確認するとわかります。このため、「window」オブジェクトのプロパティと変数名が重複してしまうと、正常に動作しなくなるので注意してください。

次のコードを実行すると、変数「a」の内容も「window」オブジェクトのプロパティ「a」の内容も同じ文字が表示されます。

```
a = "OK";
// 変数「a」の内容を表示する
alert(a);
// 「window」オブジェクトのプロパティ「a」の内容を表示する
alert(window.a);
```

▶ 変数の削除

変数を削除するには、「delete」演算子を使います(117ページ参照)。ただし、「var」宣言した変数を削除することはできません。

Ⅲ JavaScriptでの定数の定義

「var」を使って宣言した変数の内容は、自由に変更できます。内容を変更してほしくない場合には「var」でなく、「const」を使います。「const」を使って宣言する場合には、定数名(変数名)と同時に値も設定する必要があります。定数名の制限などは、変数と同様です。なお、「const」はIE9(Windows Phone含む)では動作しません。

```
// constで定数として宣言
const android = 1;
// constで宣言した変数は変更ができないのでエラー
android = android + 1;
```

COLUMN　constでの不具合

Operaでは「const」宣言はできますが、変数の内容を変更することができてしまいます。

同名の変数を「try...catch」文内で「const」宣言し、「catch」ブロック内で「var」宣言した場合、Firefox/Google Chromeではエラーになりますが、Safari/Operaではエラーになりません。

```
try{
  const mac = "SE/30";
  alert("constを使用しました");
}catch(e){
  alert("constは使えないのでvarで代用します");
  var mac = "SE/30";
}
```

繰り返し処理(ループ)について

JavaScriptでの繰り返し処理の種類

JavaScriptには繰り返し処理を行う命令として、「for」文、「while」文、「do...while」文が用意されています。

「for」文を使った繰り返し処理

「for」文は、一定の回数、処理を繰り返します。「for」文の構文は、次のようになります。

```
for (初期値; 繰り返し条件; 増減値){
  繰り返す処理
}
```

たとえば、「0」から「9」までの合計を求めるには、次のように記述します。

```
n = 0;
for (i=0; i<10; i++){
  n = n +i
}
```

初期値・繰り返し条件・増減値は「,」(カンマ)で区切って複数、指定することができます。たとえば、次のように記述することができます。

```
n = 0;
m = 0;
for (i=0, j=1; i<10, j<5; i++, j++){
  n = n + i;
  m = m + j;
}
```

この場合、「j」の値が「5」以上になったら終了するので、変数「n」は「6」、変数「m」は「10」となります。

増減値を指定する場合、ポストインクリメント(i++)よりもプリインクリメント(++i)の方が処理が速い場合もあります。

また、繰り返しを途中で抜ける場合には、「break」文を使います。繰り返しの先頭に戻る場合には、「continue」文を使います。

▶ 初期値・繰り返し条件・増減値の省略

初期値・繰り返し条件・増減値のすべてを省略することが可能です。すべて省略する場合は、次のように記述します。

```
for (;;)
```

ただし、これは無限に繰り返すことになり、Web Workers上で動作させる場合以外では、ブラウザが停止状態になってしまうことがあります。このような場合には、次のように「break」文を使って繰り返し処理から抜け出すことができます(直近の繰り返し処理から抜ける)。

```
n = 0;
for (;;){
  n++;
  if (n > 10) break;
}
```

▶ サンプルでの確認

次のコードでは、「0」から「9」までの文字と、「10」から「1」までの文字が表示されます。

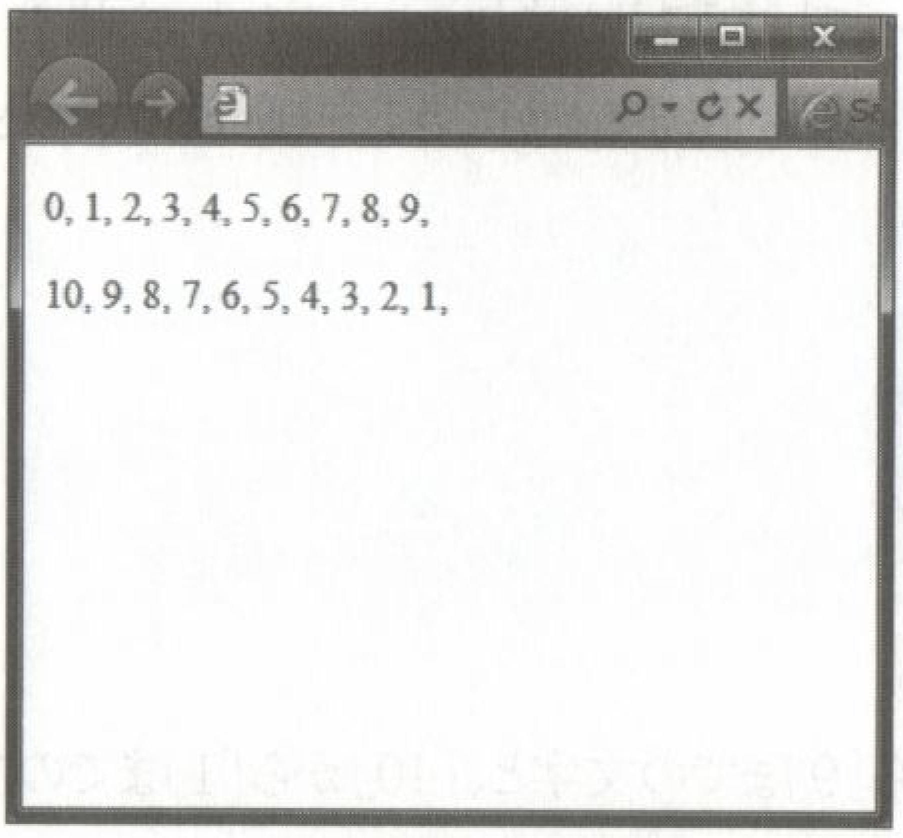
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 文字入れる変数を用意
  var total = "";
  // 10回繰り返す
  for(var i=0; i<10; i++){
    total = total + (new String(i)) + ", ";
  }
  // 結果を出力
  document.getElementsByTagName("output")[0].innerHTML = total;
  // 文字入れる変数を初期化
  total = "";
  // 10回繰り返す
  for(var i=10; i>0; i--){
    total = total + (new String(i)) + ", ";
  }
  // 結果を出力
  document.getElementsByTagName("output")[1].innerHTML = total;
}, false);
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



||| 「while」文、「do...while」文を使った繰り返し処理

条件を満たしている間、繰り返し処理を行うには、「while」文、または、「do...while」文を使います。構文はそれぞれ、次のようになります。

```
while (条件式){
  繰り返す処理
}
```

```
do{
  繰り返す処理
} while (条件式);
```

「while」文は繰り返し前に条件を調べるため、条件によっては一度も処理が行われない場合があります。「do...while」文の場合は最後に条件を調べるため、最低1回は繰り返し処理が行われます。また、「while」文、「do...while」文とも、「break」文を使って繰り返しから抜けることができます。

▶ サンプルでの動作確認

次のコードでは、最初のループでは10回繰り返されます。2回目のループは条件を満たしていない場合でも最低1回は実行されるため、「100」の数値が表示されます。

SAMPLE CODE HTMLのコード(index.html)

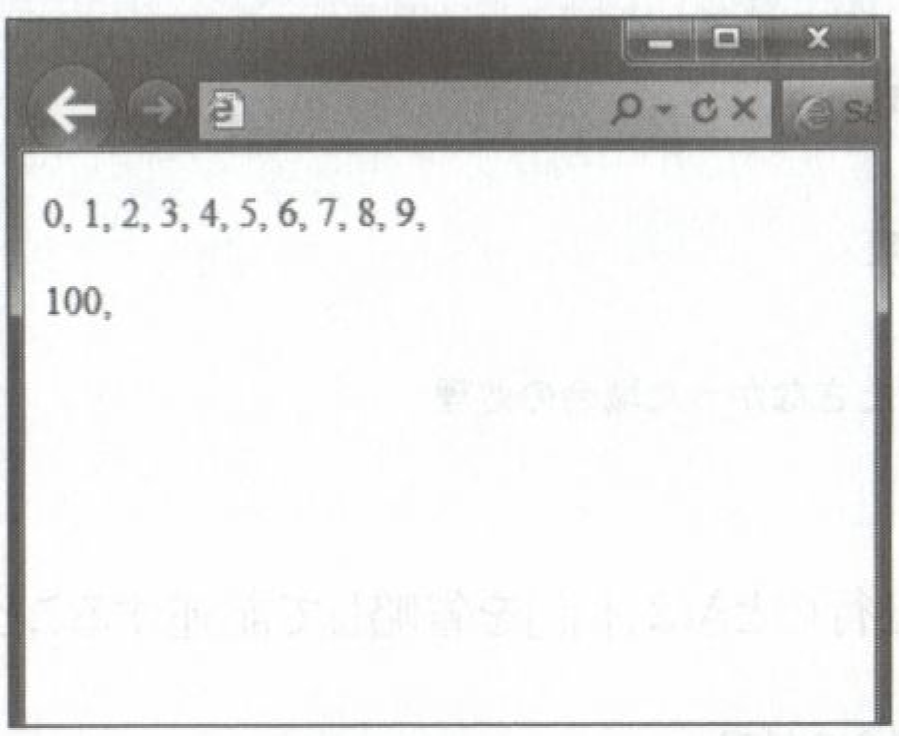
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
```

```
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 文字を入れる変数を用意
  var total = "";
  // 10回繰り返す
  var count = 0;
  while(count <10){
    total = total + (new String(count)) + ", ";
    count++;
  }
  // 結果を出力
  document.getElementsByTagName("output")[0].innerHTML = total;
  // 文字を入れる変数を用意
  total = "";
  // 繰り返す。do...whileは条件判断が最後なので最低1回は実行される
  count = 100;
  do{
    total = total + (new String(count)) + ", ";
    count++;
  }while(count <10)
  // 結果を出力
  document.getElementsByTagName("output")[1].innerHTML = total;
}, false);
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



COLUMN DOMを繰り返し操作する場合

DOMのノードを繰り返し操作する場合は注意が必要です。DOMのノードを削除するような場合には、動的にノードが変化します。このとき、DOMノードの先頭から処理するとうまくいかない場合があります。このような場合にはDOMノードの末尾から処理するようにします。

条件分岐について

条件に応じて処理を分けるには

JavaScriptで条件に応じて処理を分けるには、「if」文、「switch」文、三項演算子を使います。

「if」文

「if」文は、条件を満たした場合に、以後に続く処理を行います。条件を満たしていない場合の処理も同時に記述するには、「else」節を使います。複数の条件で分岐させたり、ブロック内でさらに「if」を記述することもできますが、あまり条件判断が複雑になるようであれば、関数やメソッドなどでまとめるなどの工夫が必要になります。「if」文の構文は、次のようになります。

```
if (条件) {
  条件を満たした場合の処理
}
```

```
// 条件を満たしていない場合の処理も記述する
if (条件) {
  条件を満たした場合の処理
} else {
  条件を満たさなかった場合の処理
}
```

```
// 複数の条件で分岐させる
if (条件1) {
  条件1を満たした場合の処理
} else if (条件2){
  条件2を満たした場合の処理
} else {
  条件1、条件2のどちらも満たさなかった場合の処理
}
```

なお、「if」に続く処理が1行のときは、「{}」を省略して記述することができます。

```
if (条件) 条件を満たした場合の処理
```

「switch」文

「if」文はYes/Noの判断には向いていますが、複数の条件をまとめて処理する場合には、「switch」文を使います。「case」節に一致する値や文字列を指定し、「:」(コロン)で区切って直後に処理する命令を記述します。ここで処理を終わらせる場合には、「break」文を使って「switch」から抜けます。「break」文を記述しないと、次の「case」節の処理も実行されるの

で注意してください。

「case」節で指定した以外の値でも処理を行いたい場合には、「default」節を使います。「default」節は、省略することができます。

```
switch(調べる対象){
  case 値1 : 調べる対象と値1が一致した場合の処理
      break;
  case 値2 : 調べる対象と値2が一致した場合の処理
      break;
  default : caseで指定した以外の値の場合の処理
}
```

三項演算子

条件があまり複雑ではなく、Yes/Noだけの判別ができる場合には、三項演算子を使う方法もあります。三項演算子については、43ページも参照してください。

```
条件式 ? 条件を満たした場合の処理 : 条件を満たさなかった場合の処理
```

サンプルでの動作確認

次のコードでは、ダイアログで数値を入力し、入力値された値によって処理を分岐しています。

SAMPLE CODE HTMLのコード(index.html)

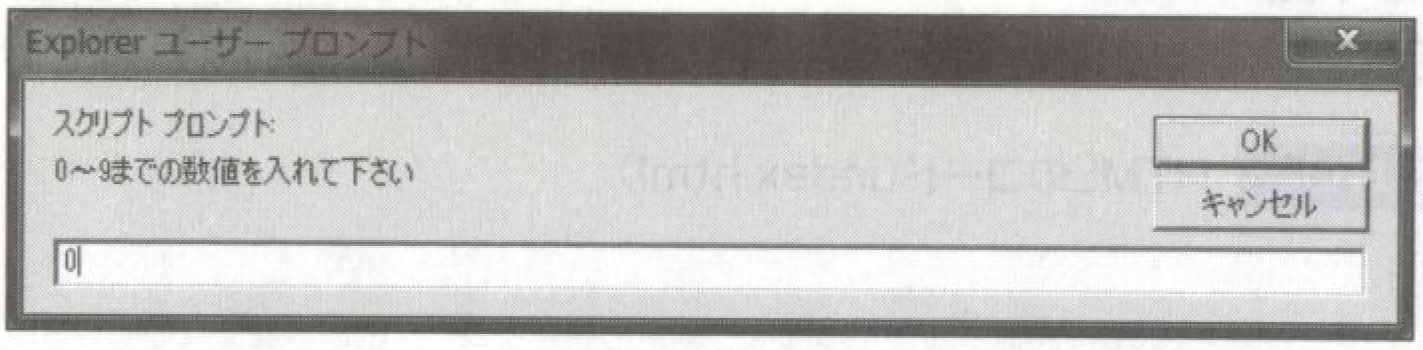
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

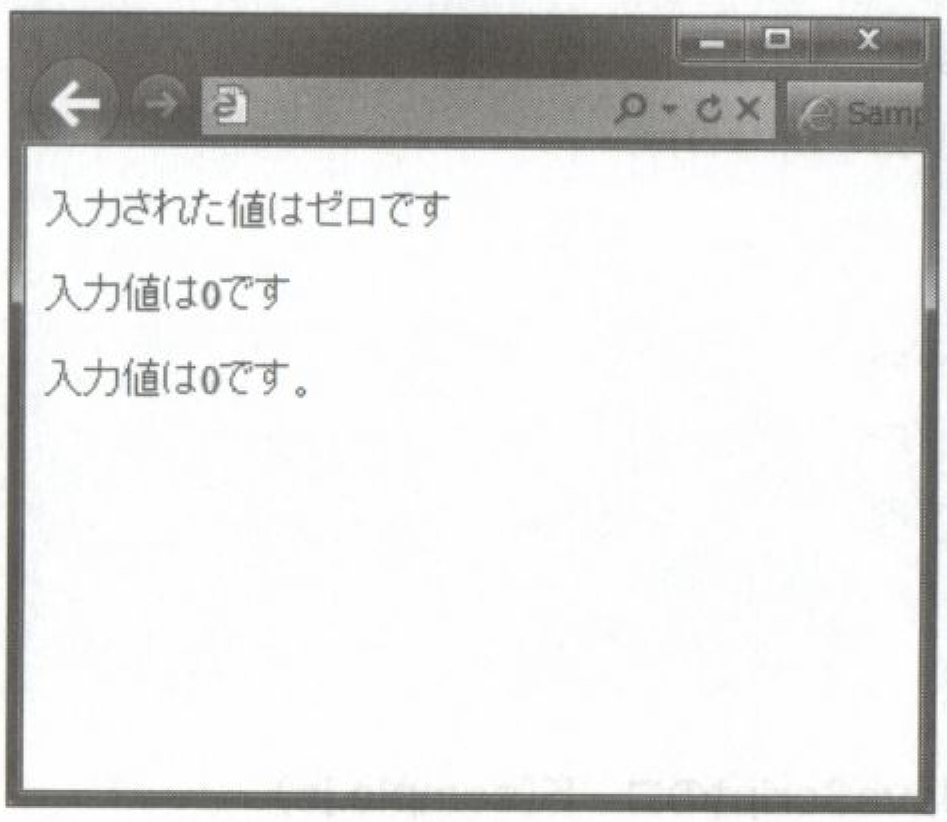
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // ダイアログで数値を入力
  var num = prompt("0～9までの数値を入れて下さい", 1) | 0;
  // 0かどうか調べる
  if (num === 0){
    ele[0].innerHTML = "入力された値はゼロです";
  }else{
```

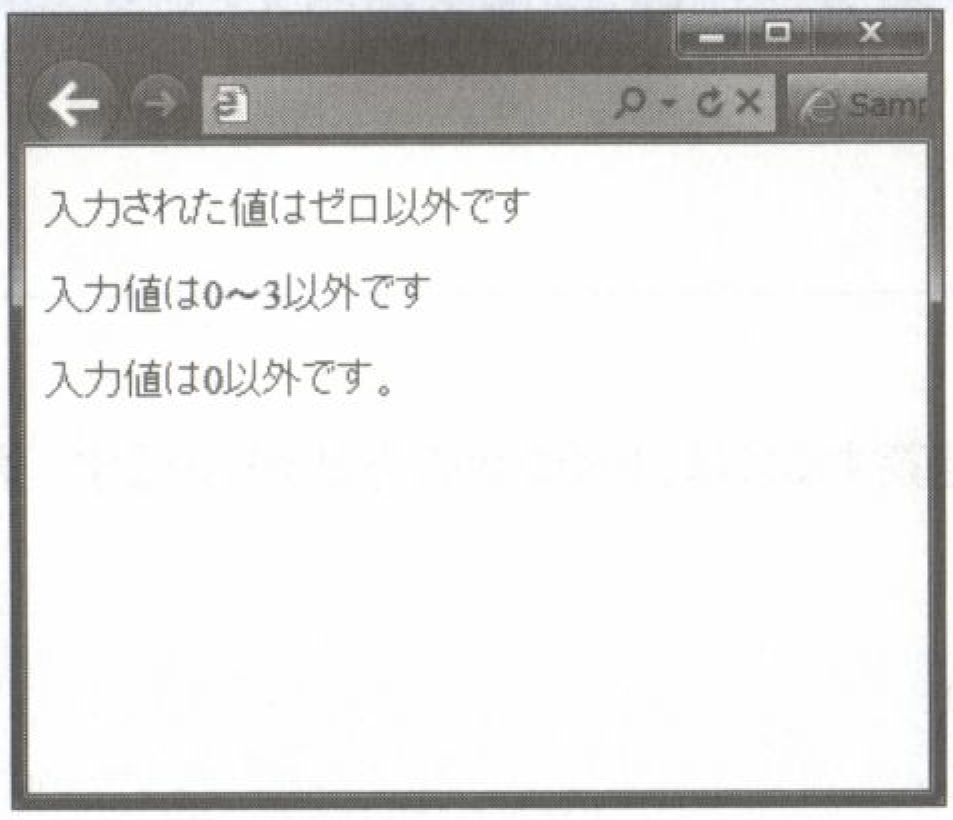
```
    ele[0].innerHTML = "入力された値はゼロ以外です";
  }
  // 入力された数値に応じて処理を分ける
  switch(num){
    case 0 : ele[1].innerHTML = "入力値は0です";
      break;
    case 1 : ele[1].innerHTML = "入力値は1です";
      break;
    case 2 : ele[1].innerHTML = "入力値は2です";
      break;
    case 3 : ele[1].innerHTML = "入力値は3です";
      break;
    default : ele[1].innerHTML = "入力値は0～3以外です";
  }
  // 三項演算子を使って処理を分ける
  ele[2].innerHTML = (num === 0) ? "入力値は0です。" : "入力値は0以外です。"
}, false);
```

ブラウザでindex.htmlを表示すると、ダイアログが表示されます。ダイアログに「0」を入力して「OK」ボタンをクリックすると、次のように表示されます。





「0」以外の数値を入力すると、次のように表示されます。



COLUMN 配列/ハッシュを利用する

サンプルのように条件に応じて文字を表示したり、特定の関数やメソッドを呼び出したりするのであれば、配列/ハッシュを利用することもできます。次の例では入力された数値に応じて配列にメッセージを入れておき、入力値を配列の添え字にすることでメッセージを表示しています。なお、配列にメッセージが入っていない範囲の場合は、次のように「||」を使って、別途、メッセージを表示することができます。

```
var ele = document.getElementsByTagName("output");
// ダイアログで数値を入力
var num = prompt("0~9までの数値を入れてください", 1) | 0;
// 配列に表示文字を代入
var msg = ["0です", "1です"];
// 0かどうか調べる
ele[0].innerHTML = msg[num] || "0と1以外です";
```

関数の定義

関数を定義する

JavaScriptで関数を定義するには、いくつかの方法があります。通常の関数定義は、次のようになります。

```
function 名前(パラメータ){
  処理
}
```

パラメータは「,」(カンマ)で区切っていくつでも記述することができます。渡すパラメータの種類に制限はありません。

JavaScriptでは変数に関数を代入することができるため、次のように定義することもできます。ただし、通常の関数定義では処理が実行される前に関数定義が行われるので、プログラム中のどこでも関数を呼び出すことができますが、変数に代入する場合は代入後でないと関数を呼び出して使用することはできません。

```
var 名前 = function(パラメータ){ 処理 }
```

イベントハンドラやメソッド、および、一時的に処理を呼び出して使用する場合には、次のように名前のない関数(無名関数/匿名関数)を使うこともできます。

```
function(パラメータ){
処理
}
```

また、次のようにして定義した関数を即時実行させることもできます(即時関数)。

```
(function(パラメータ){
処理
})(パラメータ)
```

「function」文で定義する以外にも、「new Function()」を使って行うこともできます。「new Function()」では、パラメータも処理内容も文字列で指定します。複数のパラメータを指定することができますが、パラメータの最後が実際に処理する内容になります。「new Function()」は処理速度が遅いため、あまり使われません。

```
var 名前 = new Function(パラメータ,...,処理)
```

関数内でさらに関数を定義することもできます。この場合、関数外からは関数内に定義された関数を呼び出すことはできません。

サンプルでの動作確認

次のコードでは、定義した2つの関数が呼び出され、処理が行われます。処理が行われると、画面に文字が出力されます。

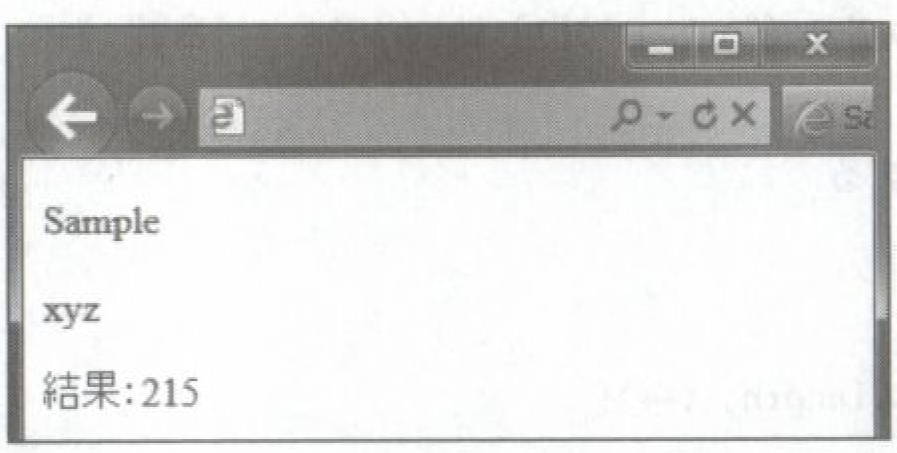
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 関数xyzを定義
  function xyz(){
    document.getElementsByTagName("output")[1].innerHTML = "xyz";
  }
  outputText("Sample");  // 関数outputTextを呼び出す
  xyz();  // 関数xyzを呼び出す
  var n = add(200, 15);  // 関数addを呼び出す
  document.getElementsByTagName("output")[2].innerHTML = "結果 :"+n;
}, false);
// 関数outputTextを定義
function outputText(text){
  document.getElementsByTagName("output")[0].innerHTML = text;
}
// 関数addを定義
var add = new Function("a", "b", "return a+b");
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



関連項目 ▶▶▶

● 関数のパラメータの読み出しについて …………………………… p.64

関数のパラメータの読み出しについて

関数のパラメータを読み出す方法

JavaScriptの関数に渡すパラメータの数は固定されておらず、自由な数だけ指定することができます。渡すパラメータの種類にも制限がありません。関数側では、渡されたパラメータは「arguments」配列を使って参照することができます。「arguments.length」が渡されたパラメータの総数になります。最初に渡されたパラメータは、「arguments[0]」で参照することができます。

サンプルでの動作の確認

次のコードでは、関数に渡された数値を合計した結果が表示されます。関数を2回、呼び出しているので、2つの計算結果が表示されます。

SAMPLE CODE HTMLのコード(index.html)

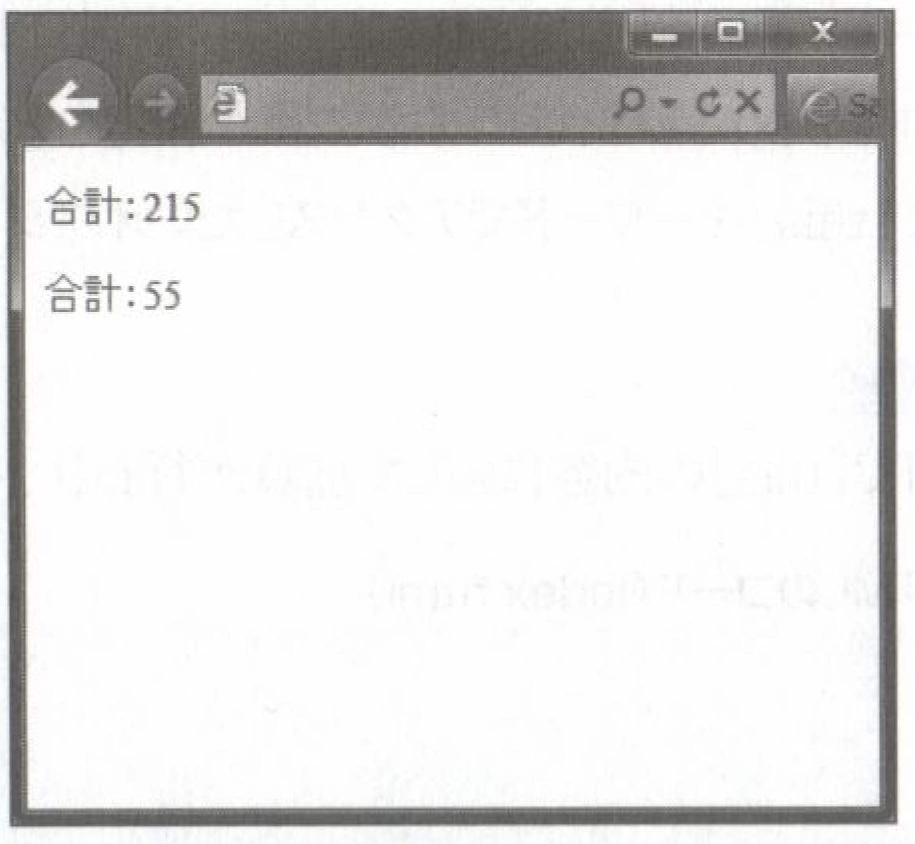
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 関数calcを呼び出し
  var n = calc(200, 15);
  document.getElementsByTagName("output")[0].innerHTML = "合計 :"+n;
  // パラメータの数を変更して呼び出し
  n = calc(1,2,3,4,5,6,7,8,9,10);
  document.getElementsByTagName("output")[1].innerHTML = "合計 :"+n;
}, false);
// パラメータの数だけ処理する
function calc(){
  var total = 0;
  for(var i=0; i<arguments.length; i++){
    var n = arguments[i];
    total = total + n;
```

```
  }
  return total;
}
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



関連項目 ▶▶▶

● 関数の定義 ………………………………………………………………… p.62

「this」キーワードの固定について

「this」キーワードを固定する方法

JavaScriptでは「this」キーワードは固定されておらず、状況によって何を示すかが変わります。この「this」キーワードを決める(バインド)するには、「bind()」メソッドを使います。「bind()」メソッドのパラメータに、「this」キーワードでアクセスしたいオブジェクトを指定します。

サンプルでの動作確認

次のコードでは、関数内で「this」の内容に応じて加算が行われ、その結果が表示されます。

SAMPLE CODE HTMLのコード(index.html)

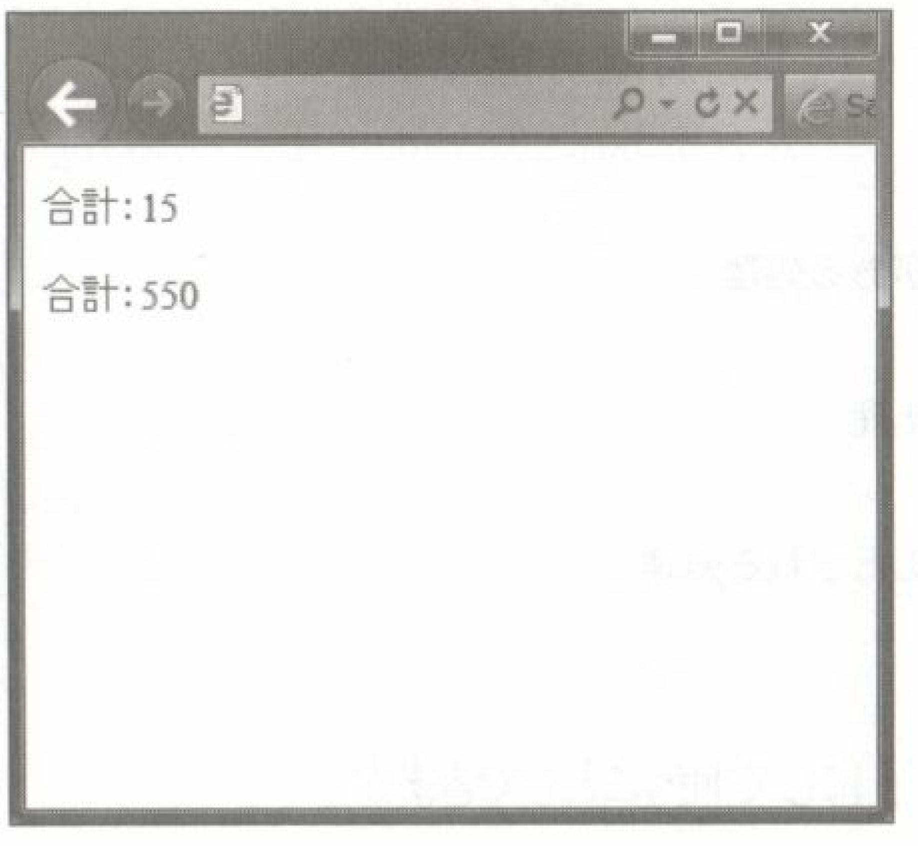
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // thisにバインド(紐付ける)配列その1
  var ary1 = [1, 2, 3, 4, 5];
  // thisにバインド(紐付ける)配列その2
  var ary2 = [10,20,30,40,50,60,70,80,90,100];
  // 関数calcにary1をバインド
  var myCalc1 = calc.bind(ary1);
  var n = myCalc1();
  document.getElementsByTagName("output")[0].innerHTML = "合計:"+n;
  // 関数calcにary2をバインド
  var myCalc2 = calc.bind(ary2);
  n = myCalc2();
  document.getElementsByTagName("output")[1].innerHTML = "合計:"+n;
}, false);
// 数値の合計を計算。bindしない場合はthisはwindowになる
function calc(){
  var total = 0;
  for(var i=0; i<this.length; i++){
```

```
    total = total + this[i];
  }
  return total;
}
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



例外処理について

JavaScriptでの例外処理

JavaScritpで例外処理を行うには、「try...catch」文を使います。構文は、次のようになります。「finally」ブロックについては、省略可能です。

```
try{
  エラーが発生する可能性がある処理
} catch(e) {
  エラーが発生した場合の処理
} finally {
  エラーの有無に関係なく実行される処理
}
```

また、次のように、入れ子にして使うこともできます。

```
try {
  エラーが発生する可能性がある処理1
  try {
    エラーが発生する可能性がある処理2
  }catch(e){
    処理2でエラーが発生した場合の処理
  }
}catch(e){
  処理1でエラーが発生した場合の処理
}
```

エラーが発生すると、エラーの原因を示す文字列が「catch」のパラメータとして渡されます。発生するエラーの種類は、次の表のようになります。

エラーの種類	説明
EvalError	「eval()」実行時のエラー/式や文などが間違っている場合など原因多数
RangeError	値が範囲外/正数または整数のみ指定するところに負数や小数値を指定している場合など
ReferenceError	参照不能/変数名が間違っている、オブジェクトがない場合などに発生
SyntaxError	文法エラー/誤字脱字等、入力ミスの可能性あり
TypeError	型が間違っている/引数の型が違う、渡したパラメータが不足して値が「null」や「undefined」などになっている場合など
URIError	「encodeURI()」「decodeURI()」で不正な引数が渡された場合
DOMException	DOMで例外発生/対象ノードがないなど

また、「throw」文を使って故意にエラーを発生させることができます。この場合、「throw」文で指定した文字列などが「catch」のパラメータとして渡されます。

さらにエラーが発生すると、「catch」にはエラーオブジェクトが渡されます。エラーオブジェクトの「message」プロパティにエラーメッセージが入ります。これは、どのブラウザでも共通で

す。Firefoxでは、これに加えて「lineNumber」プロパティにエラー発生行が入ります。また、「fileName」プロパティには、エラーとなったファイルのURLが入ります。

サンプルでの動作確認

次のコードでは、例外が発生すると、例外の原因が表示されます。2番目のエラーは故意に発生させ、任意のメッセージを表示しています。

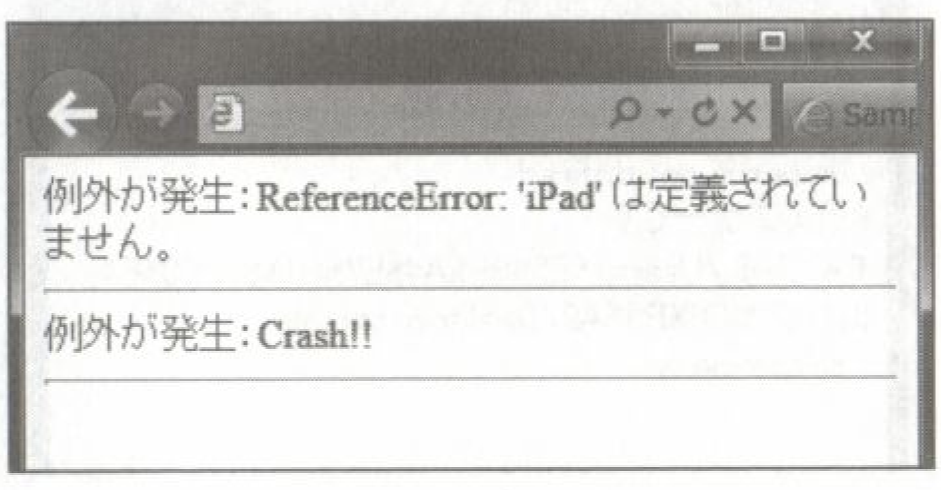
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output><hr>
    <output></output><hr>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // エラーが発生する可能性がある処理を行う
  try{
    // iPadというオブジェクトは存在しないのでエラー
    var myDevice = new iPad();
  }catch(e){
    ele[0].innerHTML = "例外が発生 :"+e;
  }
  // throwを使ってエラーを投げる
  try{
    throw "Crash!!";
  }catch(e){
    ele[1].innerHTML = "例外が発生 :"+e;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、次のように表示されます。



また、エラーオブジェクトを生成するサンプルは、次のようになります。

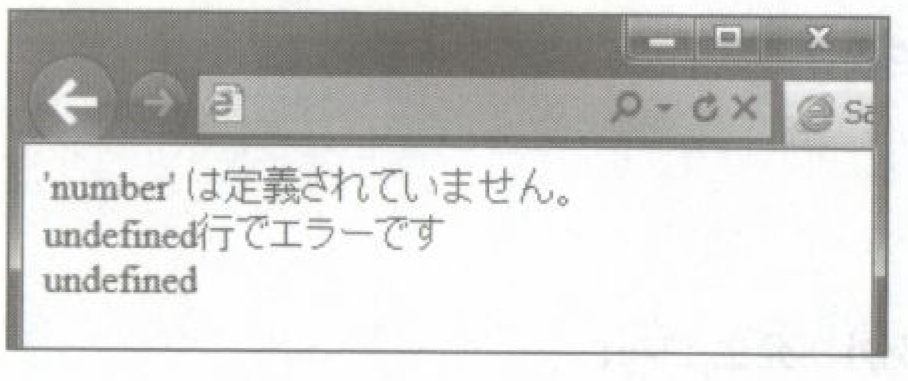
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

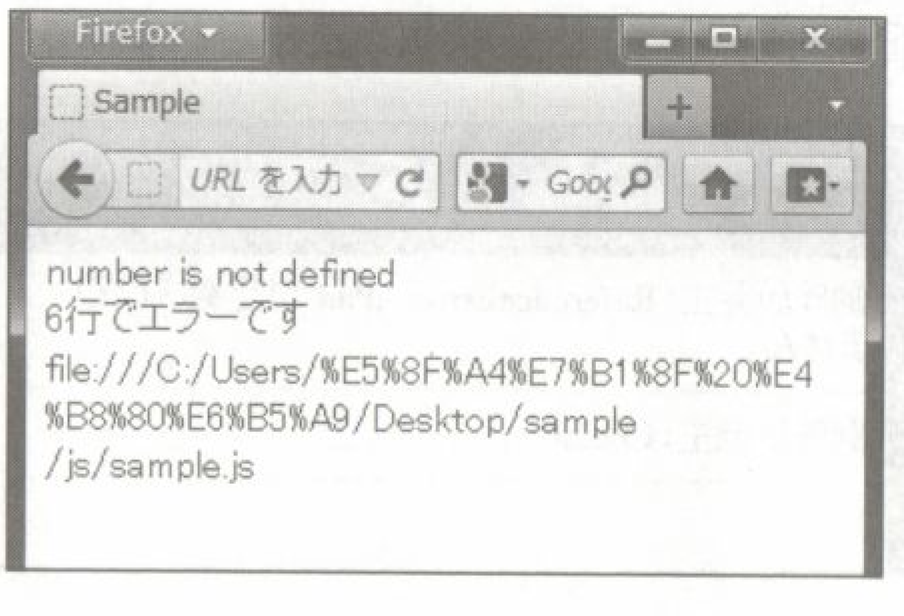
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // エラーを発生させる
  try{
    // 入力ミスによるエラー
    var n = new number(12.3);
  }catch(e){
    ele.innerHTML = e.message+"<br>";
    ele.innerHTML += e.lineNumber+"行でエラーです<br>";
    ele.innerHTML += e.fileName;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、エラーメッセージが表示されます。Firefox以外ではエラー行とエラーファイル名は「undefined」になります。



Firefoxではエラーメッセージに加えて、エラー発生行とエラーファイル名のURLが表示されます。



イベント処理について

JavaScriptのイベントの設定と解除

JavaScriptでは、「要素がクリックされた」「マウスが重なった」「ページが読み込まれた」など、さまざまなイベントがあります。主なイベントは、次の表のようになります。

イベント	説明
click	クリックした
dblclick	ダブルクリックした
mouseup	マウスの左ボタンが離された
mousedown	マウスの左ボタンが押された
mouseover	マウスが要素に重なった
mouseout	マウスが要素から離れた
keyup	キーが離された
keypress	キーが押されたまま
keydown	キーが押された
load	読み込みが完了した
scroll	スクロールした
touchstart	画面に指がタッチされた
touchmove	画面にタッチしたまま動かした
touchend	画面から指が離された

JavaScriptでイベントを設定するには、要素に用意されているプロパティにイベントハンドラ(イベント発生時に呼び出す関数)を設定します。イベントハンドラを設定できるプロパティ名には規則があり、イベント名の先頭に「on」を付加したものになります。たとえば、「click」イベントであれば、「onclick」プロパティにイベントハンドラを設定します。また、設定したイベントハンドラを解除するには、「onclick」プロパティに「null」を設定します。他のイベントも同様に設定・解除することができます。

イベントは、イベントリスナーを使って設定することもできます。イベントリスナーを使って設定する場合は、すでに設定されているイベントを上書きしません。多人数での作成や、ライブラリを作成・利用するのに便利です。イベントリスナーを設定するには、「addEventListener()」メソッドを使います。「addEventListener()」メソッドには、「イベント名」「イベントハンドラ」「イベントの伝達方向」の3つを指定します。「addEventListener()」メソッドで設定したイベントを解除するには、「removeEventListener()」メソッドを使います。ただし、「addEventListener()」メソッドで無名関数を使用した場合は、イベントを解除できないので注意が必要です。

なお、イベントについては、CHAPTER 09を参照してください。

サンプルでの動作の確認

次にコードでは、リスト項目がクリックされたら、イベントハンドラを呼び出します。最初のリスト項目は「onclick」プロパティにイベントハンドラを設定してあり、クリックするとアラートダイアログが表示されます。2番目のリスト項目は、イベントリスナーを使って2つのイベントハンドラを設定し

ています。このため、クリックすると、2回アラートダイアログが表示されます。

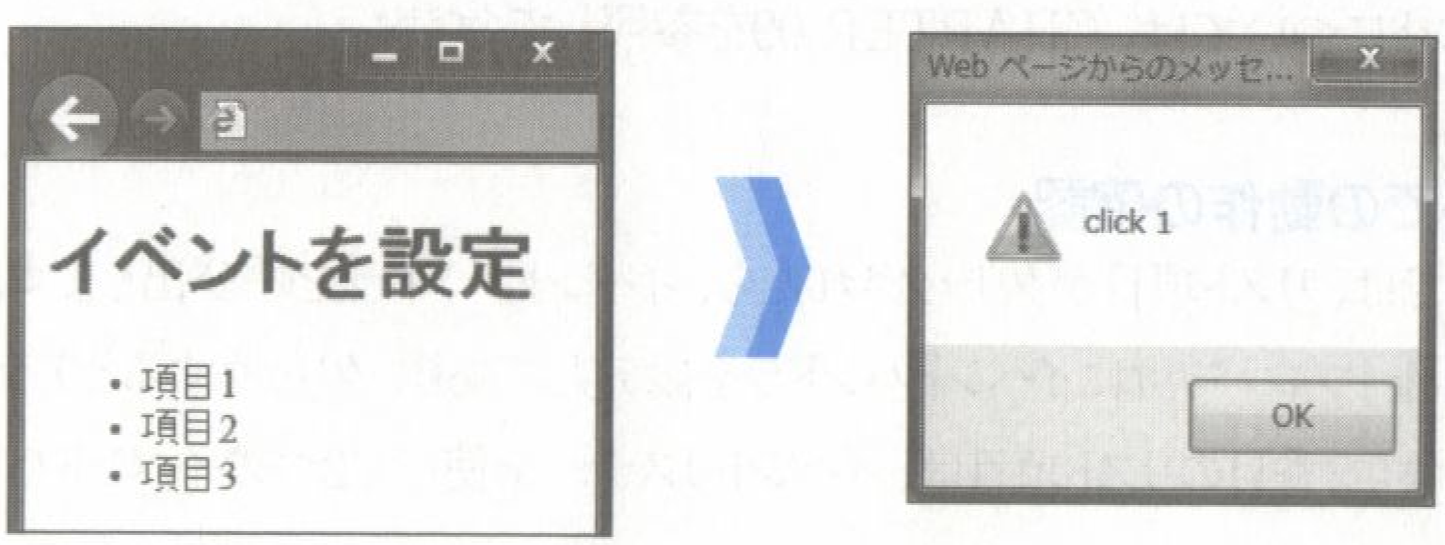
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>イベントを設定</h1>
    <ul>
      <li>項目1</li>
      <li>項目2</li>
      <li>項目3</li>
    </ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.onload = function(){
  // イベントを設定する要素を読み出し
  var ele = document.getElementsByTagName("li");
  // 最初のli要素にイベントハンドラを設定
  ele[0].onclick = function(){
    alert("click 1");
  }
  // 2番目のli要素にイベントリスナーを設定
  ele[1].addEventListener("click", function(evt){
    alert("click 2-1");
  }, true);
  // 2番目のli要素に、さらにイベントリスナーを設定
  ele[1].addEventListener("click", function(evt){
    alert("click 2-2");
  }, true);
}
```

ブラウザでindex.htmlを表示し、最初の項目をクリックすると、アラートダイアログが1回、表示されます。



2番目の項目をクリックした場合、2回アラートダイアログが表示されます。



COLUMN

IE6～8でのイベント設定

IE6～8では、「addEventListener()」「removeEventListener()」メソッドがありません。代わりに「attachEvent()」「detachEvent()」メソッドを使います。「attachEvent()」メソッド、「detachEvent()」メソッドともパラメータは同じで、最初にイベント名、2番目にイベントハンドラを指定します。イベント名は、「onclick」のように、先頭に「on」を付ける必要があります。具体的には次のようなコードになります。

```
window.onload = function(){
  // イベントを設定する要素を読み出し
  var ele = document.getElementsByTagName("li");
  // 最初のli要素にイベントハンドラを設定
  ele[0].onclick = function(){
    alert("click 1");
  }
  // 2番目のli要素にイベントリスナーを設定
  ele[1].attachEvent("onclick", function(evt){
    alert("click 2-1");
  });
  // 2番目のli要素に、さらにイベントリスナーを設定
  ele[1].attachEvent("onclick", function(evt){
    alert("click 2-2");
  });
}
```

なお、「addEventListener()」メソッドの場合は登録した順番にイベントハンドラが呼び出されますが、「attachEvent()」イベントの場合は最後に登録したイベントハンドラから呼び出されます(「addEventListener()」メソッドとは逆の順番になる)。

SECTION-015

ドキュメントオブジェクトモデルについて

JavaScriptでのHTML文書/XML文書の扱い

JavaScriptでは、ブラウザ上に表示されているHTMLの要素やXMLデータ内にある要素に対してアクセスし、処理する方法が用意されています。Webブラウザでは、HTML/XML文書はドキュメントオブジェクトモデル(DOM:Document Object Model)を採用しており、JavaScriptでもドキュメントオブジェクトモデルに従った文書を操作するためのメソッドやプロパティが用意されています。主なメソッドやプロパティは、次の表のようになります。

ドキュメントオブジェクトモデルでは、HTML/XML文書の階層を保ったまま処理を行います。このため、複雑な構造や深い階層の場合などでは、処理速度が低下することがあります。

メソッド	説明
getElementById()	指定したID名の要素にアクセスする
getElementsByTagName()	指定した要素名を持つHTML/XML要素にアクセスする。結果は配列で返される
getElementsByName()	指定した名前を持つHTML/XML要素にアクセスする。結果は配列で返される
querySelector()	指定されたCSSセレクタに一致するHTML/XML要素にアクセスする
querySelectorAll()	指定されたCSSセレクタに一致するHTML/XML要素にアクセスする。結果は配列で返される
createElement()	指定された要素を生成する
createTextNode()	テキストノードを生成する
appendChild()	子ノードとして末尾に追加する

プロパティ	説明
firstChild	最初の子ノード
lastChild	最後の子ノード
nextSibling	次の兄弟要素
previousSibling	前の兄弟要素
nodeType	ノードの種類
nodeValue	ノードの値

なお、ドキュメントオブジェクトモデル(DOM)については、CHAPTER 08を参照してください。

サンプルでの動作確認

次のコードでは、最初のリスト項目に関する情報が表示されます。また、ノードを生成し、4番目のリスト項目として「li」要素を追加しています。

SAMPLE CODE HTML(index.html)

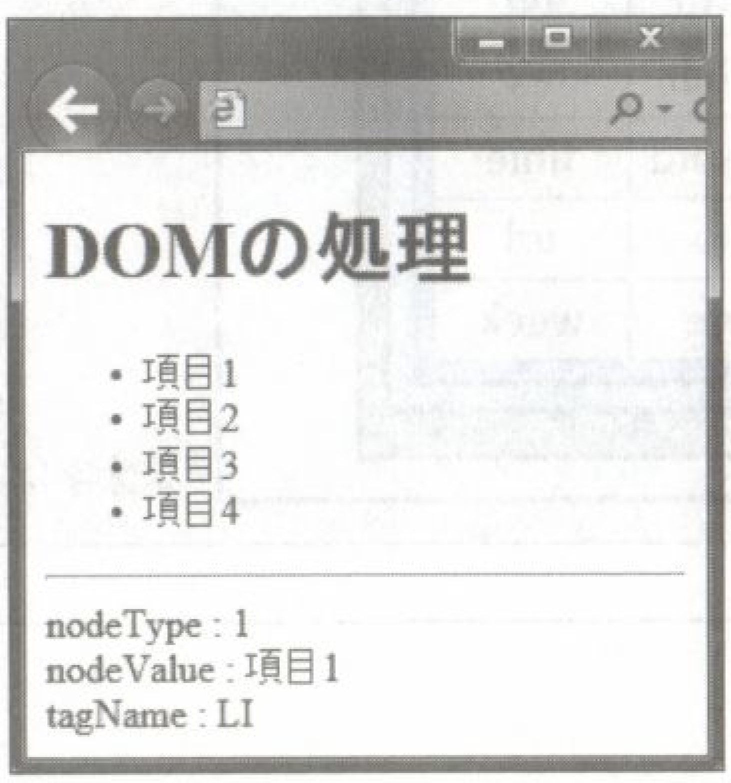
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
```

```
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>DOMの処理</h1>
    <ul>
      <li>項目1</li>
      <li>項目2</li>
      <li>項目3</li>
    </ul>
    <hr>
    <output id="result"></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

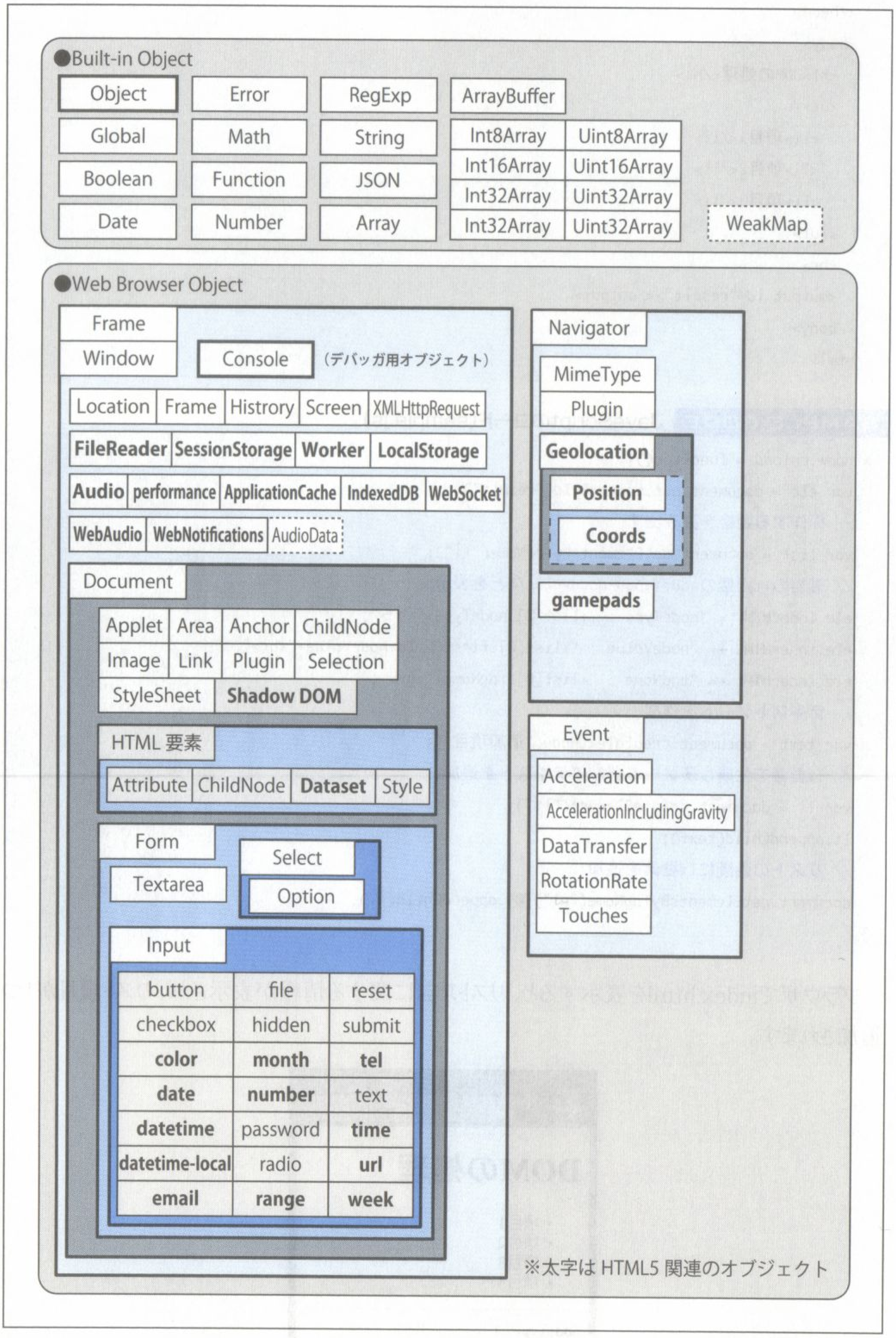
```
window.onload = function(){
  var ele = document.getElementById("result");
  // 操作する要素を読み出す
  var list = document.getElementsByTagName("li");
  // 最初のli要素のnodeTypeやnodeValueなどを表示
  ele.innerHTML = "nodeType : "+list[0].nodeType+"<br>";
  ele.innerHTML += "nodeValue : "+list[0].firstChild.nodeValue+"<br>";
  ele.innerHTML += "tagName : "+list[0].tagName+"<br>";
  // テキストノードを作成
  var text = document.createTextNode("追加項目");
  // li要素を生成し子ノードとしてテキストを追加
  var li = document.createElement("li");
  li.appendChild(text);
  // リストの最後にli要素を追加
  document.getElementsByTagName("ul")[0].appendChild(li);
}
```

ブラウザでindex.htmlを表示すると、リスト項目に関する情報が表示され、リスト項目が1つ追加されます。



DOMの階層図

DOMの階層図は、次のようになります。



CHAPTER 02

オブジェクト

文字列やオブジェクトを評価する

ここでは、JavaScriptの文字列やオブジェクトを評価する方法について解説します。

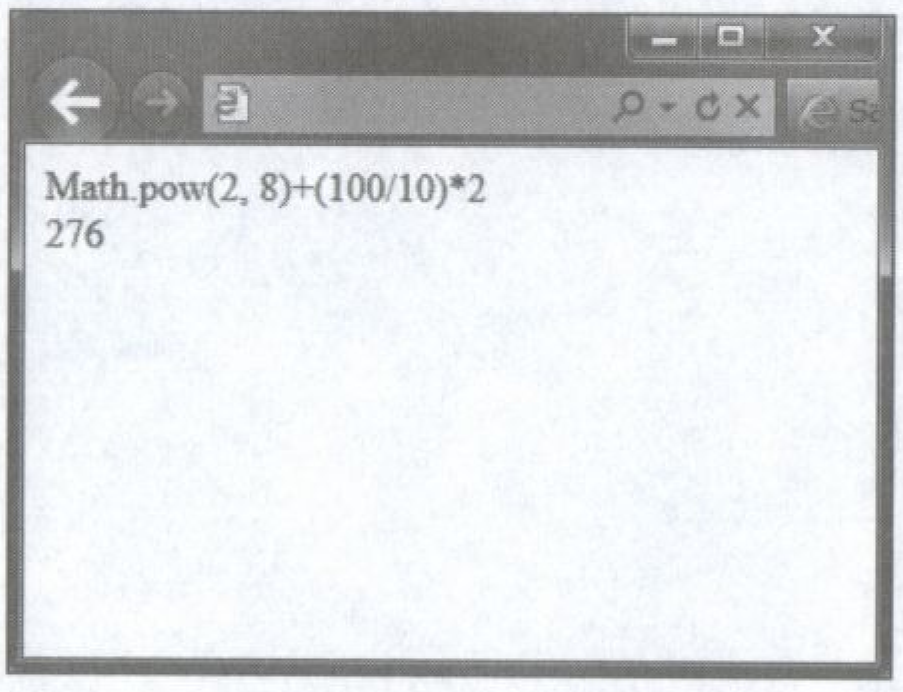
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数に評価する文字列を代入
  var text = "Math.pow(2, 8)+(100/10)*2";
  // eval()で評価
  var result = eval(text);
  // 結果を出力
  ele.innerHTML = text+"<br>"+result;
}, false);
```

ブラウザでindex.htmlを表示すると、評価する式と評価した結果が表示されます。



ONEPOINT **式や文の評価を行うには「eval()」メソッドを使う**

JavaScriptでは、オブジェクトや文字列を「eval()」メソッドを使って評価することができます。これによって、文字列をスクリプトとして解釈し、実行させることができます。「eval()」メソッドに指定できるのは評価可能なものであれば式でも文でも構いません。非同期通信を使って読み込んだJSON形式をJavaScriptのオブジェクトにする際に利用される場合もあります。ただし、この場合、不正なコードが含まれていると思わぬ不具合を発生させることがあります。JSONデータを変換する場合には、「JSON.parse()」メソッドが使えるのであれば、これを使用する方がよいでしょう。

COLUMN **ビルトインオブジェクト**

ECMA-262 5th(ECMA Script)では、次の表のビルトインオブジェクトが定義されています。ブラウザではビルトインオブジェクトに加えて、「window」や「document」など、非常に多くのオブジェクトがあります。

ビルトインオブジェクト	説明
Global Object	グローバルオブジェクト
Object	オブジェクト
Function	関数
Array	配列
String	文字列
Boolean	真偽値
Number	数値
Math	各種計算
Date	日付
RegExp	正規表現
Error	エラー
JSON	JSON(ECMA-262 5thより追加)

SECTION-017

数値かどうか調べる

ここでは、数値かどうか調べる方法について解説します。

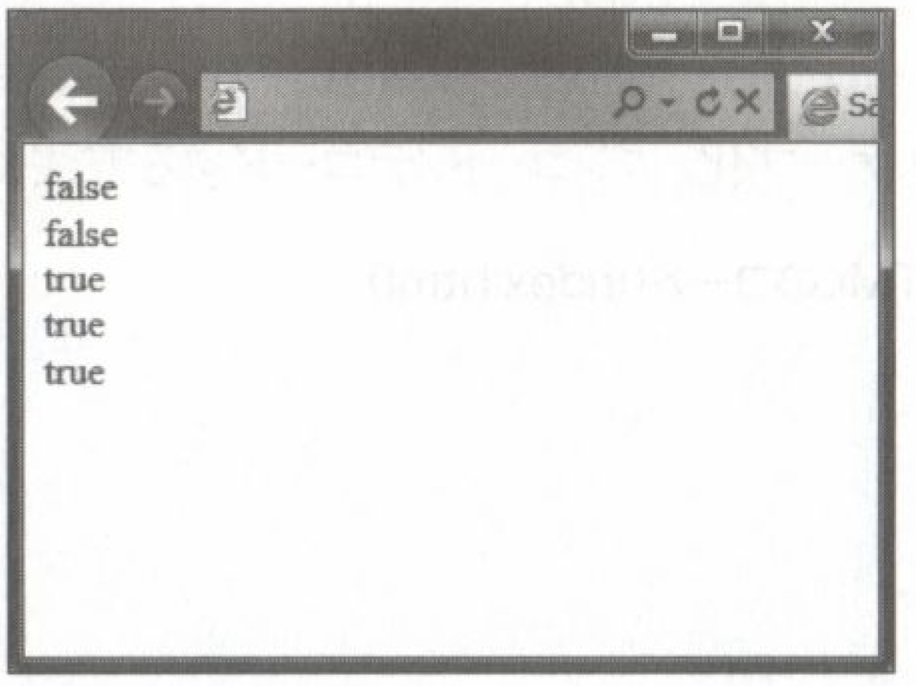
SAMPLE CODE HTMLのコードのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数に小数値を文字列として代入
  var data1 = "1969.215";
  // 変数に数値を代入
  var data2 = 2035.92;
  // 変数に文字列を変換
  var data3 = "JavaScript";
  // data1が数値以外かどうか調べる
  var flag1 = isNaN(data1);
  // data2が数値以外かどうか調べる
  var flag2 = isNaN(data2);
  // data3が数値以外かどうか調べる
  var flag3 = isNaN(data3);
  // windowオブジェクトが数値以外かどうか調べる
  var flag4 = isNaN(window);
  // 未定義のプロパティを調べる
  var flag5 = isNaN(window.evol);
  // 結果を出力
  ele.innerHTML = flag1+"<br>"+flag2+"<br>"+flag3+"<br>"+flag4+"<br>"+flag5;
}, false);
```

ブラウザでindex.htmlを表示すると、変数やオブジェクトが数値以外かどうか調べた結果が表示されます。



ONEPOINT **数値かどうかを調べるには「isNaN()」メソッドを使う**

JavaScriptでは、数値かどうかを調べるには、「isNaN()」メソッドを使います。パラメータに指定された内容が数値に変換することができなかった場合は「true」を、数値に変換できた場合は「false」を返します。数字で表現される文字列の場合、数値に変換できるのであれば、サンプルのように「false」となります。

COLUMN **整数値かどうか調べる**

「0」〜「9」までの数値だけで構成されているか、つまり、「+」記号がない整数値かどうか調べるには、次のように正規表現を使います。

```
var ele = document.getElementsByTagName("output")[0];
// 変数に数値を文字列として代入
var numStr = "1969215";
// 正規表現を使って調べる
var data = numStr.match(/\D/g);
// 結果を出力
if (data === null){
  ele.innerHTML = "0〜9の数値です";
}else{
  ele.innerHTML = "0〜9の数値ではありません";
}
```

SECTION-018

文字列をエスケープ化/エンコード/デコードする

ここでは、文字列をエスケープ化/エンコード/デコードする方法について解説します。

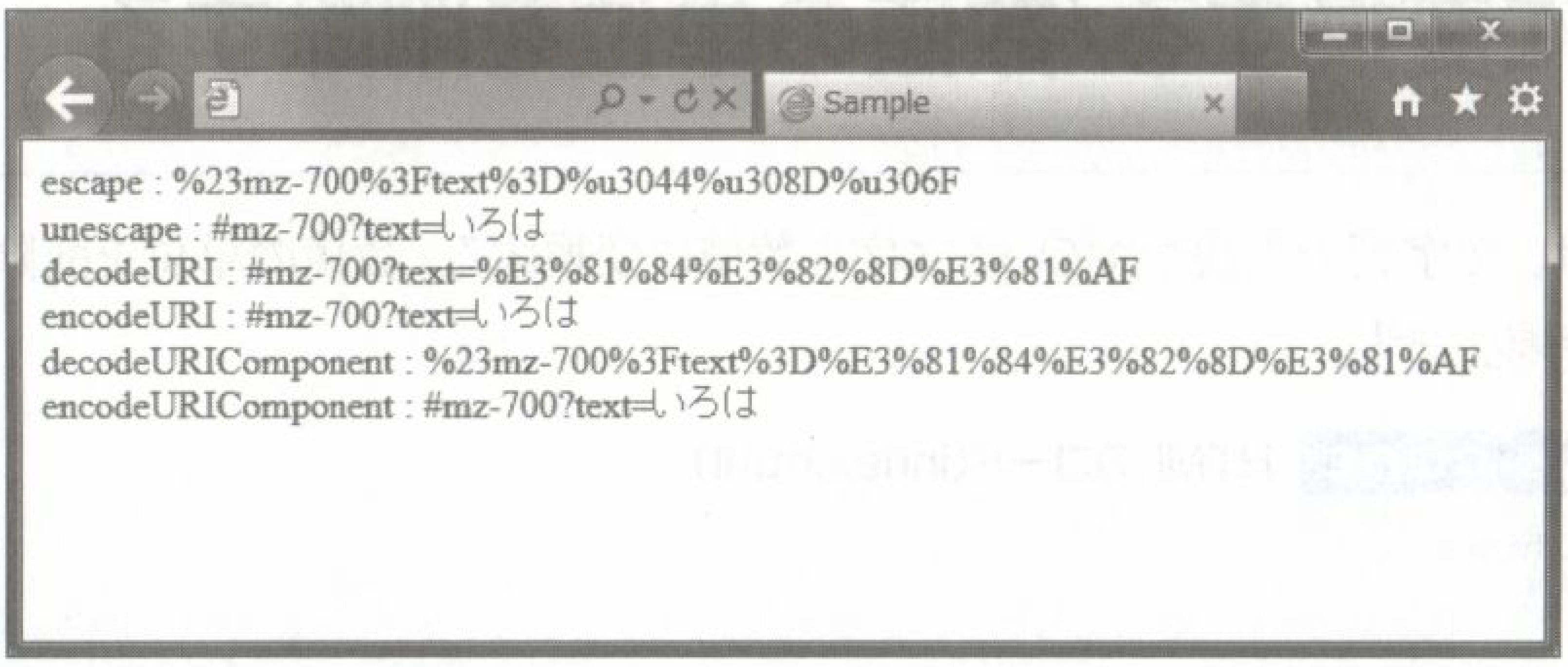
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数に文字を代入
  var text = "#mz-700?text=いろは";
  // エスケープ文字に変換
  var str1 = escape(text);
  // エスケープされた文字から変換
  var str2 = unescape(str1);
  // 文字をエンコードする
  var str3 = encodeURI(text);
  // エンコードされた文字をデコードする
  var str4 = decodeURI(str3);
  // 文字をエンコードする
  var str5 = encodeURIComponent(text);
  // エンコードされた文字をデコードする
  var str6 = decodeURIComponent(str5);
  // 結果を出力
  ele.innerHTML = "escape : "+str1+"<br>unescape : "+str2+"<br>";
  ele.innerHTML += "decodeURI : "+str3+"<br>encodeURI : "+str4+"<br>";
  ele.innerHTML += "decodeURIComponent : "+str5+"<br>encodeURIComponent : "+str6;
}, false);
```

ブラウザでindex.htmlを表示すると、文字をそれぞれ変換処理した結果が表示されます。



ONEPOINT 文字をエスケープ化するには「escape()」「encodeURI()」「encodeURIComponent()」関数を使う

サーバーにデータを送信する際に記号などは変換する必要があります。JavaScriptでは、「escape()」「encodeURI()」「encodeURIComponent()」の3種類の関数で文字をエンコード(符号化)できます。これらの関数でエンコードした文字列を元に戻す(デコード/復号化)するには、「unescape()」「decodeURI()」「decodeURIComponent()」関数を使います。それぞれの関数でどの文字が変換対象になるかは、次のようになります。なお、左上はスペースを表しています。

●「escape()」で変換される文字

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

●「encodeURI()」で変換される文字

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

●「encodeURIComponent()」で変換される文字

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

(網掛け) %xx形式に変換される文字
(白) 変換されない文字

数値に変換する/数値の単位を削除する

ここでは、文字列で表現されている文字を数値に変換する、数値の単位を削除する方法について解説します。

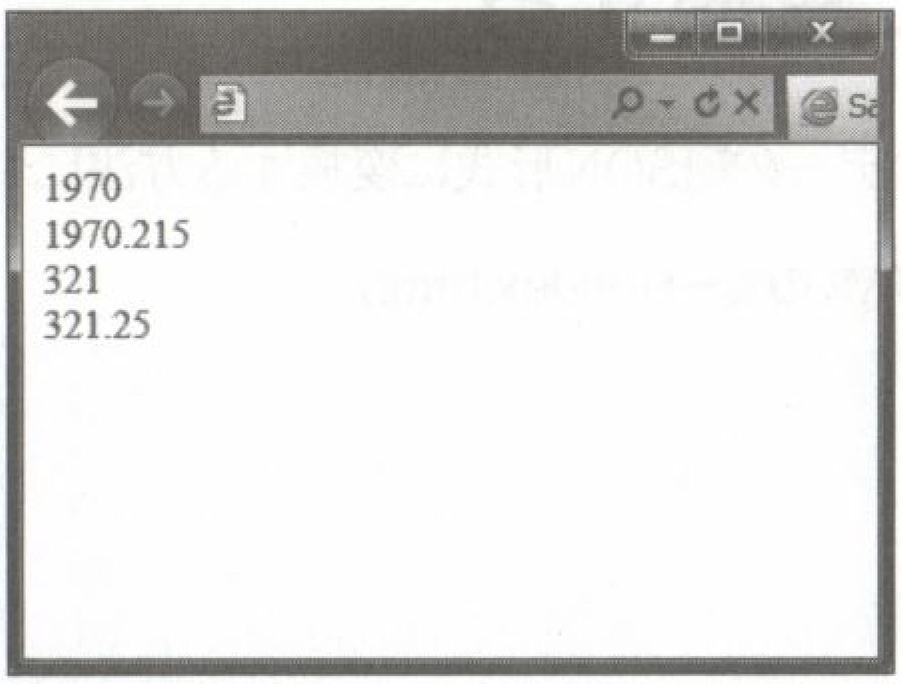
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数に小数値を文字列として代入
  var num1 = "1969.215";
  // 変数に単位付きの値を文字型として代入
  var num2 = "320.25px";
  // num1を整数値に変換
  var n1 = parseInt(num1);
  n1 = n1 + 1;  // 1を加算
  // num1を小数値に変換
  var n2 = parseFloat(num1);
  n2 = n2 + 1;  // 1を加算
  // num2を整数値に変換
  var n3= parseInt(num2);
  n3 = n3 + 1;
  // num2を小数値に変換
  var n4= parseFloat(num2);
  n4 = n4 + 1;
  // 結果を出力
  ele.innerHTML = n1+"<br>"+n2+"<br>"+n3+"<br>"+n4;
}, false);
```

ブラウザでindex.htmlを表示すると、文字で表現されている値を数値に変換し、演算した結果が表示されます。



ONEPOINT

文字を数値に変換するには「parseInt()」メソッドや「parseFloat()」メソッドを使う

JavaScriptでは変数に代入されている内容が数字を示していても、それが文字型の場合には通常の算術演算が行われません。たとえば、変数「n」に「"12"」という文字型で表現されている数値がある場合、「+」記号を使って「3」を加算すると、「15」ではなく「123」となってしまいます。これは文字列同士の連結になってしまっているためです。このような場合、文字型の数値を数値型に変換する必要があります。文字型になっている数値を数値型に変換するメソッドとして、「parseInt()」メソッドと「parseFloat()」メソッドがあります。「parseInt()」メソッドは整数値に、「parseFloat()」メソッドは小数値までを含めて変換した結果を数値型として返します。

また、CSSに関するプロパティを読み出した際、「"320px"」のように単位が付随している場合があります。座標値を読み出して演算したい場合には、この単位が邪魔になります。そこで「parseInt()」メソッドや「parseFloat()」メソッドを使えば、この単位部分だけを削除することができます。

なお、文字を数値に変換するには、「"1234" - 0;」のように、「0」を減算する方法もあります。

COLUMN

「eval()」メソッドで文字を数値に変換する

文字を数値に変換するには、「parseInt()」メソッドや「parseFloat()」メソッドを利用する以外でも、式評価を行う「eval()」メソッドを利用する方法もあります。「eval("12.3")」のようにパラメータを指定すると式の評価(演算)が行われ、「12.3」という数値になります。また、「eval("12.3+2*Math.PI")」のように計算式を指定することもできます。「eval()」メソッドは、JavaScriptで処理できるものであれば、計算式でもメソッドでもプロパティでも評価・実行が行われます。ただし、「eval()」メソッドは処理速度が速くないので高速に処理する場合は使用を避けた方がよいでしょう。また、パラメータによっては、セキュリティホールの原因になることもあります。

JavaScriptのデータをJSON形式に変換する

ここでは、JavaScriptのデータをJSON形式に変換する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

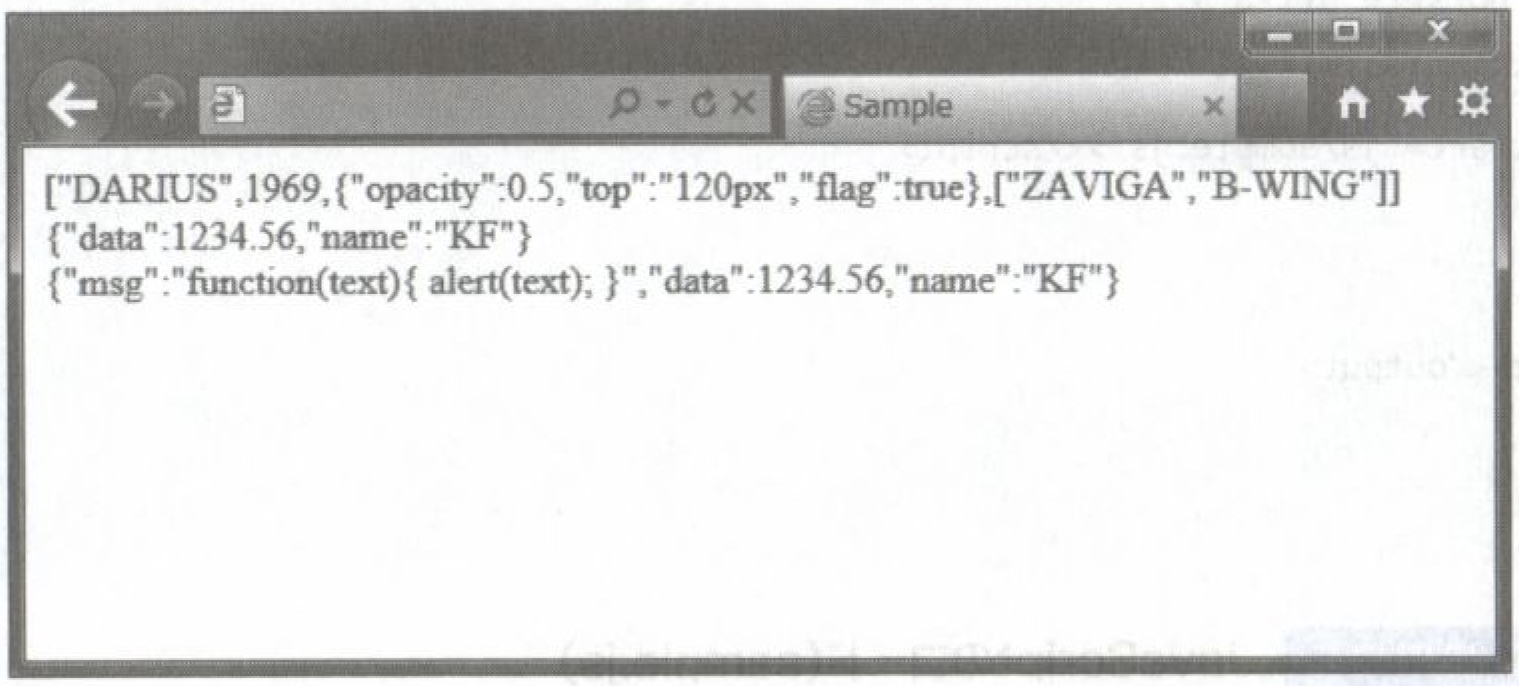
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // JSON形式に変換する例として配列を作成
  var myArray = ["DARIUS", 1969,
    {  opacity: 0.5,
      top: "120px",
      flag : true
    },
    ["ZAVIGA", "B-WING"]
  ];
  // JSON形式に変換
  var jsondata = JSON.stringify(myArray);
  // 結果を出力
  ele.innerHTML = jsondata+"<br>";
  // オブジェクト／メソッドを含む
  var myObject = {
    msg : function(text){ alert(text); },
    data : 1234.56,
    name : "KF"
  };
  // JSON形式に変換
  var jsondata2 = JSON.stringify(myObject);
  // 結果を出力
  ele.innerHTML += jsondata2+"<br>";
  // 自前で独自形式に変換
  var jsondata3 = JSON.stringify(myObject, function(key, value){
```

```
    if (new String(value).indexOf("function") > -1) {
      return new String(value);
    }
    return value;
  });
  // 結果を出力
  ele.innerHTML += jsondata3+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると。JSON形式に変換した結果が表示されます。



ONEPOINT JavaScriptのデータをJSON形式に変換するには「JSON.stringify()」メソッドを使う

JSONは、JavaScriptで扱える軽量なデータフォーマットです。JavaScriptのデータをJSON形式に変換するには、「JSON.stringify()」メソッドを使います。関数やオブジェクト、メソッドが含まれる場合、期待通りにJSON形式に変換されない場合があります。このような場合、「JSON.stringify()」メソッドの2番目のパラメータに関数を指定することで独自に変換処理することができます。この関数には「key」と「value」が渡されるので、それらに応じて処理することでサンプルのようにメソッドも文字列に変換することができます。ただし、オブジェクトでなく、配列の場合は要素ごとに「key」と「value」が渡されず、一括して関数に渡されます。

関連項目 ▶▶▶

● JSON形式からオブジェクトに変換する ………………………… p.88

JSON形式からオブジェクトに変換する

ここでは、JSON形式からJavaScriptのオブジェクトに変換する方法について解説します。

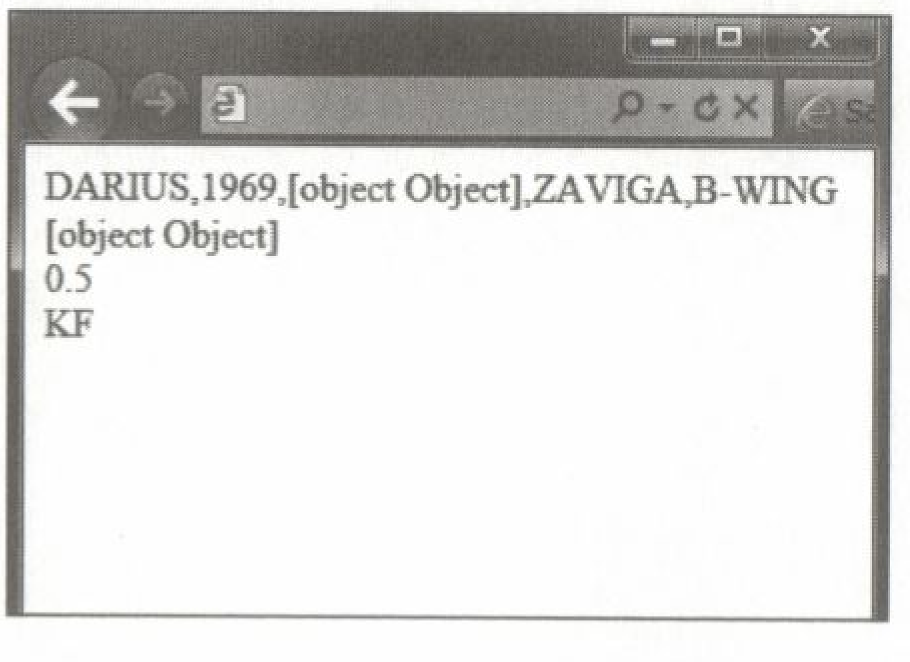
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // JSONデータを変数に代入(非同期通信で読み込む方式でも可)
  var data1 =
  '["DARIUS",1969,{"opacity":0.5,"top":"120px","flag":true},["ZAVIGA","B-WING"]]';
  var data2 = '{"data":1234.56,"name":"KF"}';
  // JSON形式から変換
  var obj1 = JSON.parse(data1);
  var obj2 = JSON.parse(data2);
  // 結果を出力
  ele.innerHTML = obj1+"<br>"+obj2+"<br>";
  // 内容を読み出して出力
  ele.innerHTML += obj1[2].opacity+"<br>";
  ele.innerHTML += obj2.name;
}, false);
```

ブラウザでindex.htmlを表示すると。JSON形式から変換して読み出した内容などが表示されます。



ONEPOINT **JSON形式からオブジェクトにするには「JSON.parse()」メソッドを使う**

非同期通信などを使ってサーバーから読み込まれたJSON形式をJavaScriptのオブジェクト形式に変換するには、「JSON.parse()」メソッドを使います。「eval()」メソッドでも変換することはできますが、「JSON.parse()」メソッドの方が、より安全に変換することができます。

また、jQueryライブラリなどでは、あらかじめJSON形式かどうか調べてから処理を行うようになっているものもあります。

COLUMN **Local StorageとJSON**

データをローカルディスク上に保存する場合に多く利用されるのが、Local Storageです。このLocal Storageにデータを保存する場合、文字列で保存する必要があります。配列や各種オブジェクトは、そのままでは保存できません。このため、配列オブジェクトなどは、「JSON.stringify()」メソッドを使って保存する必要があります。「JSON.stringify()」メソッドを使って保存した配列オブジェクトなどは、「JSON.parse()」メソッドを使って元に戻すことができます。なお、「window」オブジェクトなどは、「JSON.stringify()」メソッドを使って変換することはできません。また、「RegExp」オブジェクトも「JSON.stringify()」メソッドを使うと、空のオブジェクトである「{}」に変換されます。

関連項目 ▶▶▶

- JavaScriptのデータをJSON形式に変換する ………… p.86
- Web Storageにデータを設定する ………… p.738

SECTION-022

オブジェクトを生成する

ここでは、オブジェクトを生成する方法について解説します。

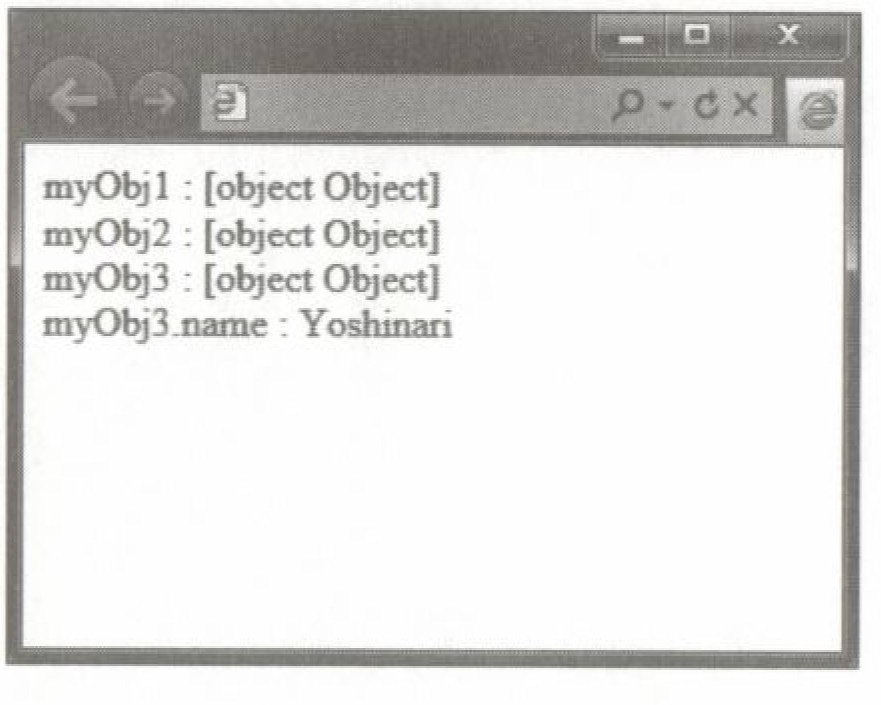
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オブジェクトを生成
  var myObj1 = new Object();
  // オブジェクトを作成
  var myObj2 = { };
  // オブジェクトを作成
  var myObj3 = { name : "Yoshinari" };
  // 結果を出力
  ele.innerHTML = "myObj1 : "+myObj1.toString();
  ele.innerHTML += "<br>myObj2 : "+myObj2.toString();
  ele.innerHTML += "<br>myObj3 : "+myObj3.toString();
  ele.innerHTML += "<br>myObj3.name : "+myObj3.name;
}, false);
```

ブラウザでindex.htmlを表示すると、3つオブジェクトを生成した結果が表示されます。最終行ではオブジェクトのプロパティの内容が表示されています。



ONEPOINT オブジェクトを生成するには「new Object()」とするか「{ }」を使う

JavaScriptでオブジェクトを生成するには、「new Object()」とするか、「{ }」を使います。「{ }」内には「キー:値」のペアを「,」(カンマ)で区切って複数、列記することができます。この場合、キー名がプロパティ、または、メソッド名になります。

独自のオブジェクトではなく、既存のオブジェクト(StringやArray)を生成したい場合は、「new」演算子を使います。

COLUMN オブジェクトの生成速度

JavaScriptでは、「new Object()」としてオブジェクトを生成する以外に「{ }」と記述して生成することもできます。どちらもオブジェクトが生成されることには変わりませんが、処理にかかる時間は大幅に異なります。「new Object()」とした場合と比べて「{ }」を使うと、2倍以上、高速にオブジェクトを生成することができます。

関連項目 ▶▶▶

● 指定されたオブジェクトを継承して新たなオブジェクトを生成する………………………… p.92

IE 6 Chrome Safari 5 Firefox 4 Opera 11 iOS 4 Android 2 WP

SECTION-023

指定されたオブジェクトを継承して新たなオブジェクトを生成する

ここでは、指定されたオブジェクトを継承し新たなオブジェクトを生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

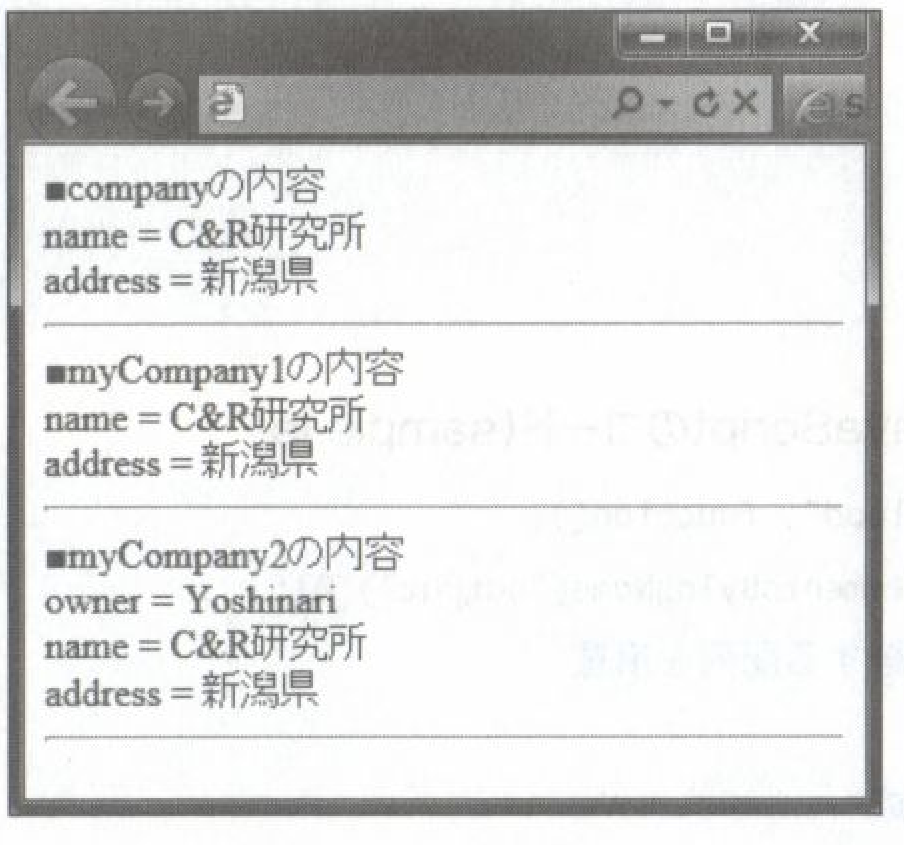
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オブジェクトを生成
  var company = { name : "C&R研究所", address : "新潟県" };
  // オブジェクトを継承して新たなオブジェクトを生成
  var myCompany1 = Object.create(company);
  // オプションを指定して生成
  var myCompany2 = Object.create(company, {
    owner : {
        value : "Yoshinari",  // プロパティの内容
        writable : false,  // 書き込み可能かどうか
        enumerable : true,  // 列挙可能。for...inで出る
        configurable : false  // 再定義可能かどうか
    },
    stype : {
        value : "男性",  // プロパティの内容
        writable : false,  // 書き込み可能かどうか
        enumerable : false,  // 列挙不可。for...inで出ない
        configurable : true  // 再定義可能かどうか
    }
  });
  // 結果を出力
  outObj(company, "company");
  outObj(myCompany1, "myCompany1");
  outObj(myCompany2, "myCompany2");
```

```
    // オブジェクトの内容を出力する関数
    function outObj(obj, text){
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML += "■"+text+"の内容<br>";
      for(var i in obj){
        ele.innerHTML += i + " = " + obj[i] + "<br>";
      }
      ele.innerHTML += "<hr>";
    }
}, false);
```

ブラウザでindex.htmlを表示すると、生成したオブジェクトのプロパティの内容が表示されます。



ONEPOINT オブジェクトを継承し新たなオブジェクトを生成するには「Object.create()」メソッドを使う

JavaScriptでオブジェクトを継承して、新たなオブジェクトを生成するには、「Object.create()」メソッドを使います。最初のパラメータには継承元となるオブジェクトを指定します。2番目のパラメータはオプションで、プロパティに対して次の表の属性を設定することができます。なお、同時に設定する属性の設定値によってはエラーとなる場合があります。

属性値	説明
value	プロパティの内容
writable	プロパティに書き込みができるかどうか。「true」なら書き込み可能、「false」なら不可
enumerable	列挙可能かどうか。「for...in」文での抽出対象になるかどうか。「true」なら列挙可能、「false」なら列挙不可
configurable	再定義可能かどうか。「true」なら再定義可能、「false」なら再定義不可
set	セッター。プロパティに値を書き込む時に実行する関数
get	ゲッター。プロパティから内容を読み出す時に実行する関数

関連項目 ▶▶▶

● オブジェクトを生成する …… p.90

オブジェクトのインスタンスを調べる

ここでは、オブジェクトのインスタンスを調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

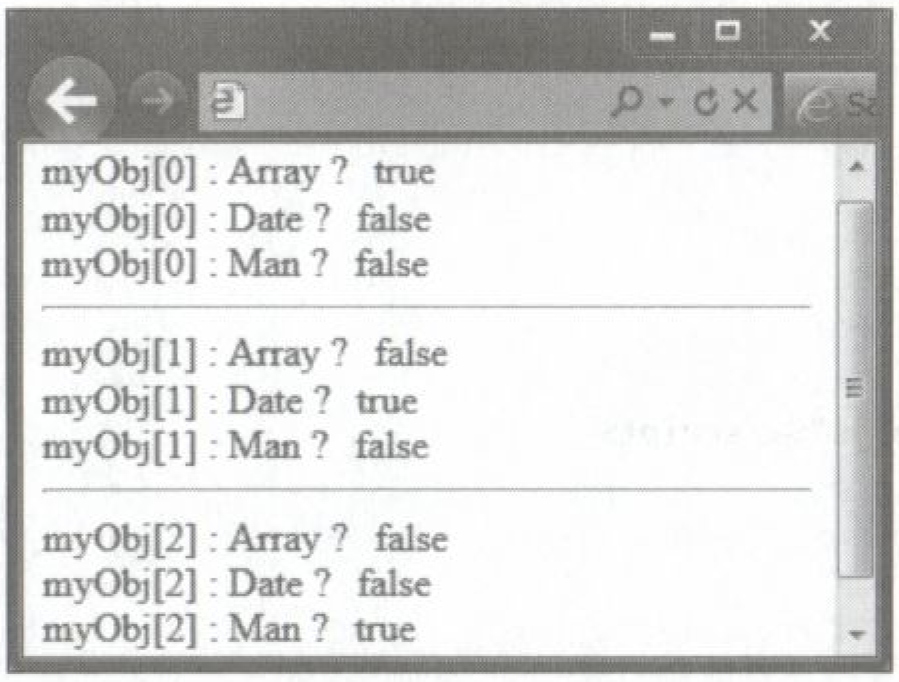
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 各種オブジェクトを格納する配列を用意
  var myObj = [];
  // 配列オブジェクトを作成
  myObj[0] = new Array();
  // Dateオブジェクトを作成
  myObj[1]= new Date();
  // manオブジェクトを作成
  myObj[2] = new Man("KF", 43);
  // 特定のオブジェクトのインスタンスか調べ結果を出力
  for(var i=0; i<myObj.length; i++){
    // 配列オブジェクトのインスタンスかどうか調べる
    var flag1 = myObj[i] instanceof Array;
    // Dateオブジェクトのインスタンスかどうか調べる
    var flag2 = myObj[i] instanceof Date;
    // 自前のオブジェクトのインスタンスかどうか調べる
    var flag3 = myObj[i] instanceof Man;
    // 結果を出力
    ele.innerHTML += "myObj["+i+"] : Array ?  "+flag1+"<br>";
    ele.innerHTML += "myObj["+i+"] : Date ?  "+flag2+"<br>";
    ele.innerHTML += "myObj["+i+"] : Man ?  "+flag3+"<hr>";
  }
  // 自前のオブジェクトを生成するためのクラス
  function Man(name,age){
    this.name = name;
```

```
    this.age = age;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、オブジェクトがどのインスタンスなのかを調べた結果が表示されます。



ONEPOINT どのオブジェクトから派生したのか調べるには「instanceof」演算子を使う

JavaScriptでは、多くをオブジェクトとして扱います。あらかじめ決められたオブジェクトから生成する以外に、独自に作成したオブジェクトを生成することもできます。生成されたオブジェクトが、どのような種類のオブジェクトなのかを調べるには、「instanceof」演算子を使います。「typeof」演算子では型が返るだけで、正確な情報はわかりません。また、「instanceof」演算子以外には、オブジェクトの「constructor」プロパティを使って生成元のオブジェクト(内容)を確認することもできます。

関連項目 ▶▶▶

● 指定されたオブジェクトを継承して新たなオブジェクトを生成する …… p.92
● プロパティのオーナーかどうか調べる …… p.114

SECTION-025

オブジェクトの内容を読み出す

ここでは、オブジェクトの内容を読み出す方法について解説します。

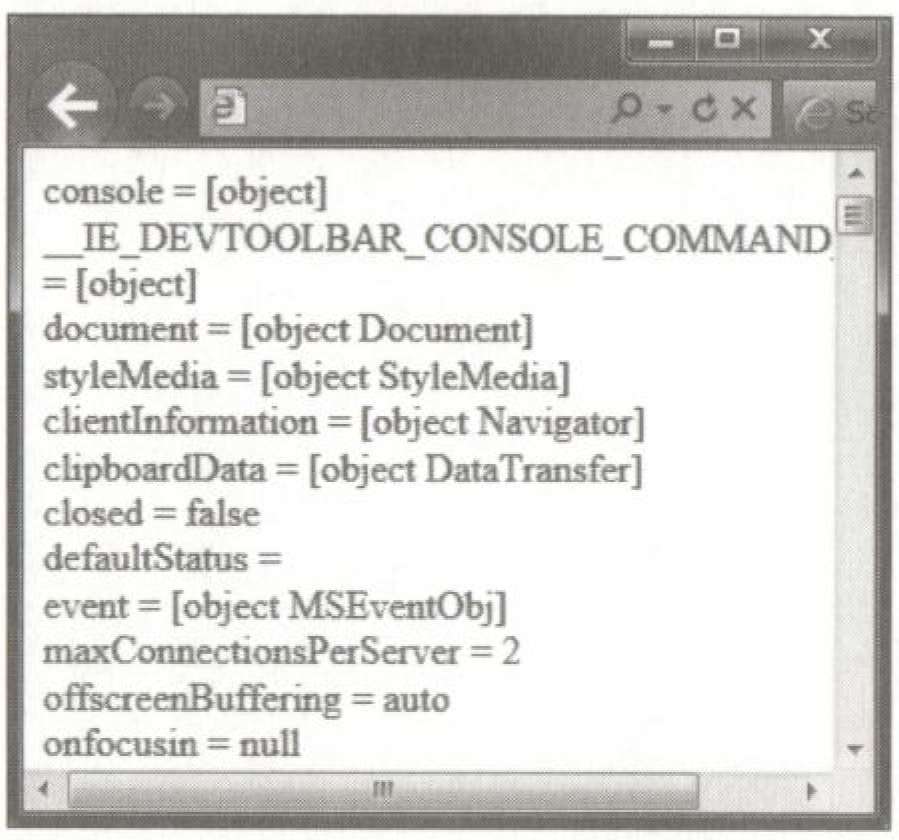
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 結果を入れる変数を用意
  var data = "";
  // windowオブジェクトの内容を出力
  for(var i in window){
    try{
      data = data + i + " = " + window[i] + "<br>";
    }catch(e){}
  }
  // 結果を出力
  document.getElementsByTagName("output")[0].innerHTML = data;
}, false);
```

ブラウザでindex.htmlを表示すると、「window」オブジェクトの内容が表示されます。



ONEPOINT オブジェクトの内容を読み出すには「for...in」文を使う

オブジェクトの内容を読み出すには、「for...in」文を使います。「for(変数名 in オブジェクト)」とすると、オブジェクトのプロパティ名が1つずつ読み出されます。「for...in」文では、オブジェクトで「DontEnum」属性(非列挙型)になっているものは読み出すことができません。「DontEnum」属性は、「Object.create()」メソッドによってオブジェクト生成するときや、「defineProperty()」メソッドや「defineProperties()」メソッドを使って後から設定することができます。これらのメソッドなどでは、次の表に示すオブジェクトの属性値を設定することができます。オブジェクトのプロパティが列挙型かどうかは、「enumerable」属性で指定することができます。

また、ブラウザによってはオブジェクト内容を読み出すとエラーになる場合があります。これは、「try...catch」文で回避することができます。

オブジェクトの属性値	説明
writable	プロパティに書き込みができるかどうか。「true」なら書き込み可能、「false」なら不可
enumerable	列挙可能かどうか。「for...in」文での抽出対象になるかどうか。「true」なら列挙可能、「false」なら列挙不可
configurable	再定義可能かどうか。「true」なら再定義可能、「false」なら再定義不可

COLUMN 「in」演算子を使ってプロパティがあるか調べる

「in」演算子を使うと、オブジェクトにプロパティがあるかどうか調べることができます。調べることができるのはプロパティとオブジェクトで、メソッドは対象外になります。次の例では、「navigator」オブジェクトに「geolocation」が含まれる場合は「true」、そうでない場合は「false」が変数「flag」に入ります。

```
var flag = "geolocation" in navigator;
```

なお、「in」演算子は、プロトタイプチェーンをたどってプロパティがあるかどうかを調べます。プロパティがそのオブジェクトだけに存在するものかどうか(オーナーかどうか)を調べるには、「hasOwnProperty()」メソッドを使います。

関連項目 ▶▶▶

- オブジェクトのプロパティの読み出し・書き込み処理を設定する …… p.102
- オブジェクトのプロパティ一覧を取得する …… p.110
- プロパティのオーナーかどうか調べる …… p.114

SECTION-026

プロトタイプを利用する

ここでは、プロトタイプを利用する方法について解説します。

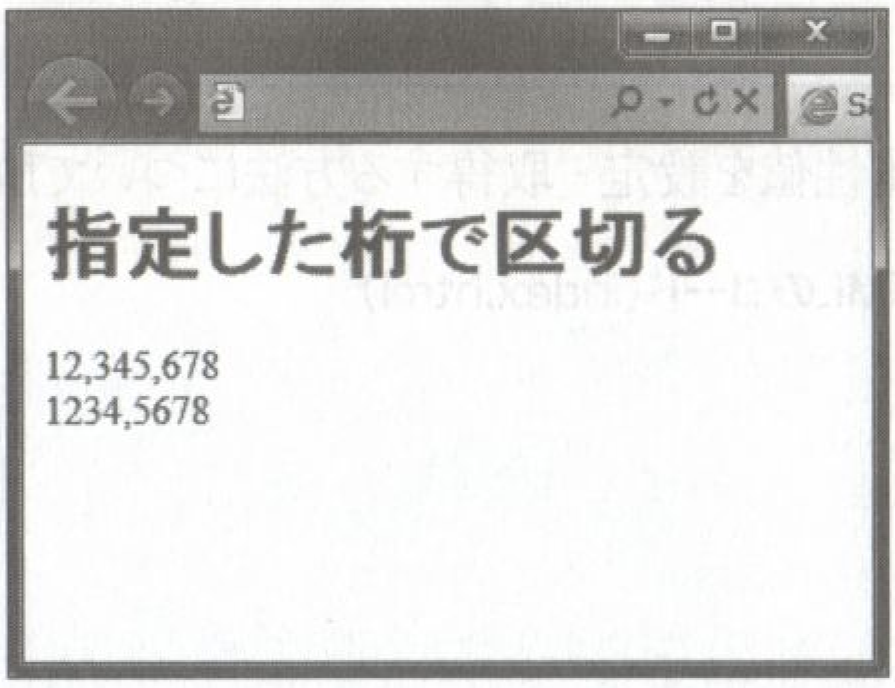
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>指定した桁で区切る</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Numberオブジェクトにn桁で区切るメソッドを追加
  Number.prototype.numFormat = function(separate){
    // パラメータが省略された場合は3桁ごとにする
    separate = separate || 3;
    // 結果を入れる
    var n = "";
    var count = 0;
    // 数値を文字列に変換
    var str = new String(this);
    for (var i=str.length-1; i>=0; i--){
      n = str.charAt(i) + n;
      count++;
      if (((count % separate) == 0) && (i != 0)) n = ","+n;
    }
    return n;
  }
  // 数値を文字列として3桁で区切る
  var n = (12345678).numFormat();
  ele.innerHTML = n+"<br>";
  // 数値を文字列として4桁で区切る
  var n = (12345678).numFormat(4);
  ele.innerHTML += n;
}, false);
```

ブラウザでindex.htmlを表示すると、「Number」オブジェクトの「prototype」に指定した桁数で区切るメソッドを追加し、そのメソッドを使って3桁と4桁に区切られます。



ONEPOINT プロトタイプを利用するには「prototype」を使う

JavaScriptではプロパティやメソッドを継承する際にプロトタイプの仕組みを利用します。プロトタイプは各オブジェクトに「prototype」という名前で用意されています。ここにプロパティやメソッド、オブジェクトを設定すると継承されたオブジェクトにも、それらが追加・定義されます。たとえば、「Object」オブジェクトの「prototype」にメソッドを追加すれば、ArrayやFunction、自前で作成したオブジェクトでも同じメソッドを利用することができるようになります。

継承されたオブジェクトで同名のプロパティやメソッドが定義された場合は、それらが優先して処理されます。

関連項目 ▶▶▶

- 指定されたオブジェクトを継承して新たなオブジェクトを生成する……………………… p.92
- プロパティのオーナーかどうか調べる ………………………………………………… p.114

SECTION-027

プロパティの属性値を設定・取得する

ここでは、プロパティの属性値を設定・取得する方法について解説します。

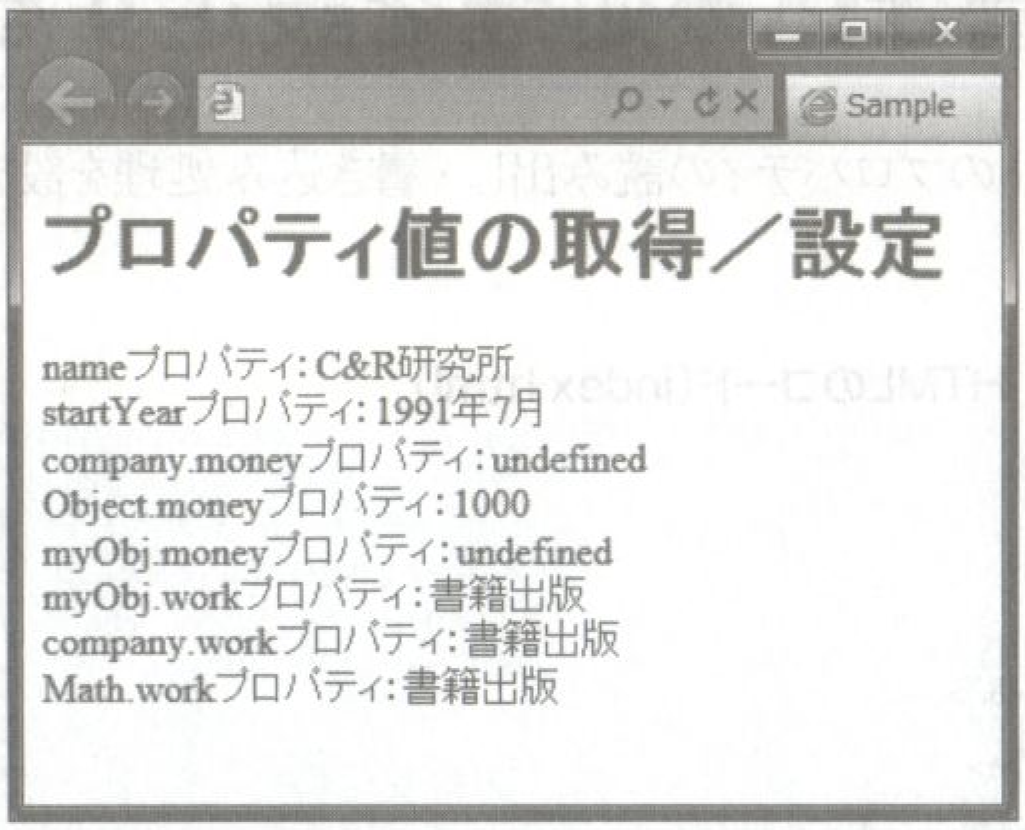
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>プロパティ値の取得／設定</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // おおもとのオブジェクトにプロパティを設定
  // Objectだけに設定されるプロパティ
  Object.money = 1000;
  // すべてのオブジェクトに継承されるプロパティ
  Object.prototype.work = "書籍出版";
  // オブジェクトを生成
  var company = { name : "C&R研究所", address : "新潟県" };
  // 新たなプロパティに書き込み
  company.startYear = "1991年7月";
  // 新規にオブジェクトを生成
  var myObj = new Object();
  // プロパティを読み出し
  ele.innerHTML = "nameプロパティ:"+company.name+"<br>";
  ele.innerHTML += "startYearプロパティ:"+company.startYear+"<br>";
  ele.innerHTML += "company.moneyプロパティ:"+company.money+"<br>";
  ele.innerHTML += "Object.moneyプロパティ:"+Object.money+"<br>";
  ele.innerHTML += "myObj.moneyプロパティ:"+myObj.money+"<br>";
  ele.innerHTML += "myObj.workプロパティ:"+myObj.work+"<br>";
  ele.innerHTML += "company.workプロパティ:"+company.work+"<br>";
  ele.innerHTML += "Math.workプロパティ:"+Math.work+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、プロパティに書き込んだ内容が表示されます。継承されたプロパティは内容が表示されますが、そうでない場合は「undefined」と表示されます。



ONEPOINT **プロパティの属性値を設定するには「オブジェクト名.プロパティ名 = 内容」、取得するには「オブジェクト名.プロパティ名」とする**

オブジェクトのプロパティに内容を設定するには、「オブジェクト名.プロパティ名 = 内容」のように、「=」を使って代入します。内容を読み出す場合には、「オブジェクト名.プロパティ名」にアクセスします。

オブジェクトに新たなプロパティを追加する場合は、「オブジェクト名.追加するプロパティ名 = 内容」のようにします。ECMA Script 5thに対応したブラウザの場合、「Object.create()」メソッド、「Object.defineProperty()」メソッド、「Object.defineProperties()」メソッドでプロパティの属性を設定することもできます。これに関しては、102ページを参照してください。

オブジェクトにプロパティを追加する際、継承先でも使用したいのであれば、「オブジェクト名.prototype.追加するプロパティ名 = 内容」のようにします。

SECTION-028

オブジェクトのプロパティの読み出し・書き込み処理を設定する

ここでは、オブジェクトのプロパティの読み出し・書き込み処理を設定するgetterとsetterについて解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>setter/getter</h1>
  </body>
</html>
```

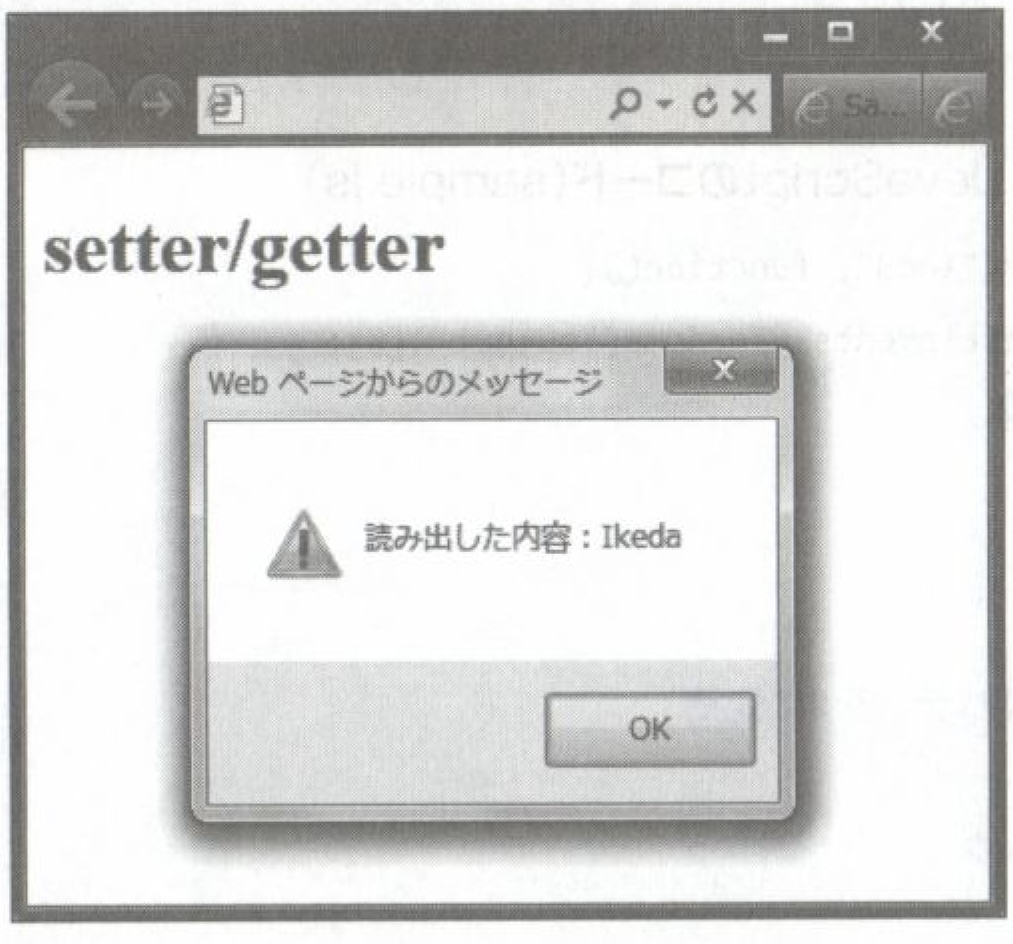
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オブジェクトを生成
  var company = { name : "C&R研究所", address : "新潟県" };
  // オプションを指定して生成
  var myCompany = Object.create(company, {
    owner : {
      enumerable : true,  // 列挙可能。for...inで出る
      configurable : true,  // 再定義可能かどうか
      set: function(value){
        alert("設定内容は「"+value+"」です");
        __value = value;  // 書き込み内容を保存
      },
      get : function(){
        alert("読み出した内容 :"+__value);
      }
    }
  });
  // オブジェクトに値を設定
  myCompany.owner = "Ikeda";
  var ownerName = myCompany.owner;
}, false);
```

ブラウザでindex.htmlを表示すると、書き込み内容がアラートダイアログに表示されます。この処理は、setter側で行っています。



その後、内容がアラートダイアログに表示されます。この処理はgetter側で行っています。



ONEPOINT **オブジェクトのプロパティの読み出し・書き込み処理を設定するには「Object.create()」「Object.defineProperty()」メソッドなどを使う**

オブジェクトのプロパティの読み出し・書き込み処理を設定するには、「Object.create()」メソッド、「Object.defineProperty()」メソッド、「Object.defineProperties()」メソッドを使います。

オブジェクトのプロパティで「set」に関数を指定すると、内容を設定する際に呼び出されます。関数には、書き込む内容が渡されます。逆に内容を読み出す場合には、「get」に関数を指定します。

関連項目 ▶▶▶

● オブジェクトの内容を読み出す …… p.96

SECTION-029

オブジェクトの変更を禁止する

ここでは、オブジェクトの変更を禁止する方法について解説します。

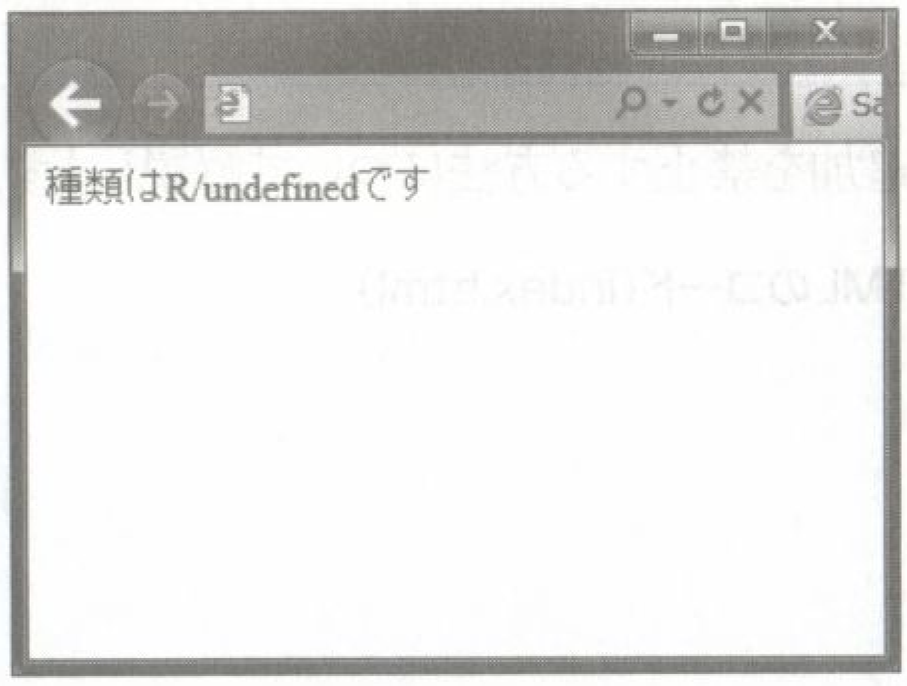
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

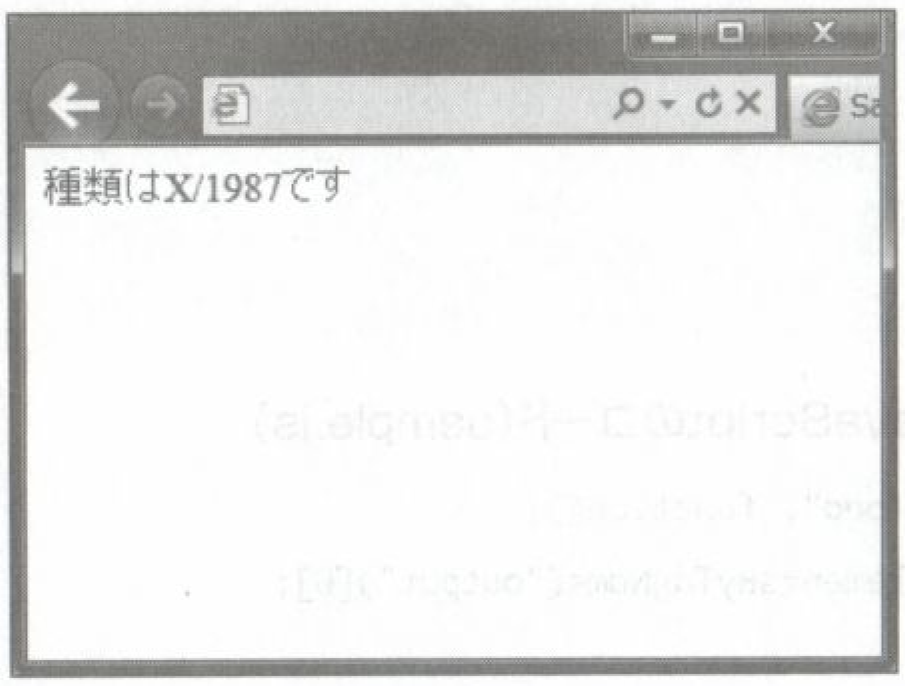
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オブジェクトを作成
  var myObj = {
    type : "N"
  };
  // オブジェクトのプロパティをRに変更
  myObj.type = "R";
  // オブジェクトをロック
  Object.freeze(myObj);
  // オブジェクトのプロパティをXに変更
  myObj.type = "X";
  // オブジェクトにプロパティを追加
  myObj.year = 1987;
  // type, yearプロパティの内容を表示する
  ele.innerHTML = "種類は" + myObj.type + "/"+myObj.year+"です";
}, false);
```

ブラウザでindex.htmlを表示すると、「freeze()」メソッドによってロックされているため、変更前のプロパティの値と「undefined」が表示されます。



なお、「freeze()」メソッドを実行しない場合は、次のような結果になります。



ONEPOINT　オブジェクトを保護するには「freeze()」メソッドを使う

JavaScriptでは、オブジェクトにプロパティやメソッドを簡単に追加できます。また、プロパティの値も自由に変更することができます。ただし、この手軽さが問題を引き起こすことがあります。そこで、オブジェクトが不用意に変更されないようにするには、「freeze()」メソッドを使います。パラメータには、内容を変更したくないオブジェクトを指定します。なお、「freeze()」メソッドの戻り値も処理したオブジェクトになります。

関連項目 ▶▶▶

●プロパティの追加を禁止する …… p.106
●オブジェクトの変更・拡張などが可能か調べる …… p.108

プロパティの追加を禁止する

ここでは、プロパティの追加を禁止する方法について解説します。

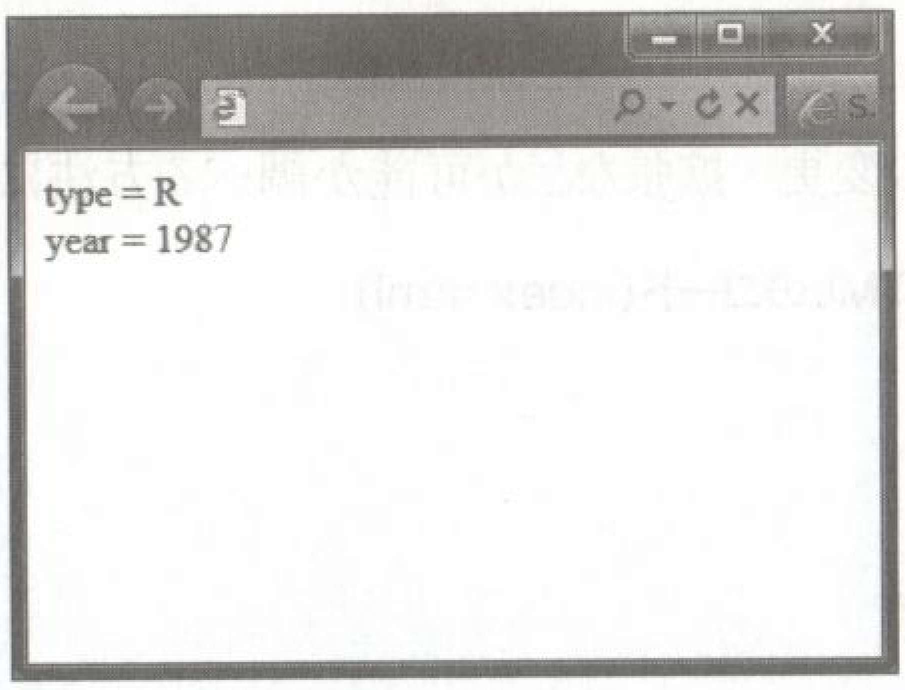
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

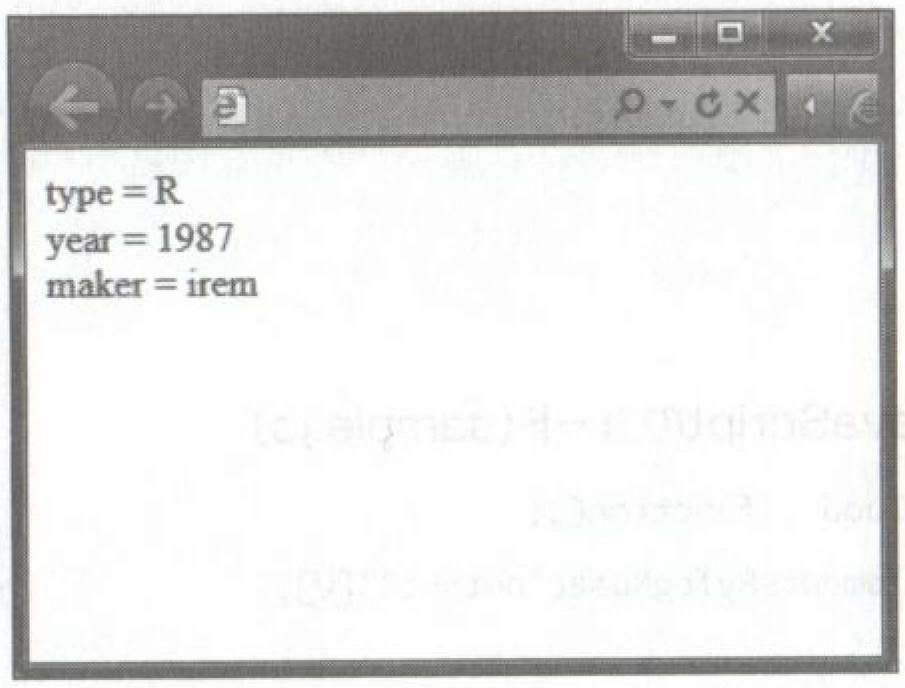
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オブジェクトを作成
  var myObj = {
    type : "N",
    code : "bug"
  };
  // オブジェクトにプロパティを追加
  myObj.year = 1987;
  // オブジェクトのプロパティの追加を禁止
  Object.preventExtensions(myObj);
  // オブジェクトにプロパティを追加
  myObj.maker = "irem";
  // オブジェクトのプロパティをRに変更
  myObj.type = "R";
  // オブジェクトのcodeプロパティを削除
  delete myObj.code;
  // オブジェクトの内容を表示する
  for(var i in myObj){
    ele.innerHTML += i + " = " + myObj[i]+"<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「preventExtensions()」メソッドによってオブジェクトへのプロパティの追加が禁止されているため、「maker」プロパティは表示されません。



なお、「preventExtensions()」を実行しない場合は、次のような結果になります。



ONEPOINT

プロパティの追加を禁止するには「preventExtensions()」メソッドを使う

JavaScriptでは、オブジェクトへのプロパティの追加を禁止するには、「preventExtensions()」メソッドを使います。パラメータには、プロパティを追加したくないオブジェクトを指定します。104ページのように、「freeze()」メソッドを使ってオブジェクト自体の変更を禁止することもできますが、これでは不便な場合もあります。オブジェクトの新たなプロパティの追加だけを禁止したい場合には、「preventExtensions()」メソッドの方が便利です。「preventExtensions()」メソッドを実行するとプロパティの追加だけが禁止され、プロパティの内容の変更や削除は自由に行うことができます。

関連項目 ▶▶▶

●オブジェクトの変更を禁止する ……………………………… p.104
●オブジェクトの変更・拡張などが可能か調べる ……………………… p.108

SECTION-031

オブジェクトの変更・拡張などが可能か調べる

ここでは、オブジェクトの変更・拡張などが可能か調べる方法について解説します。

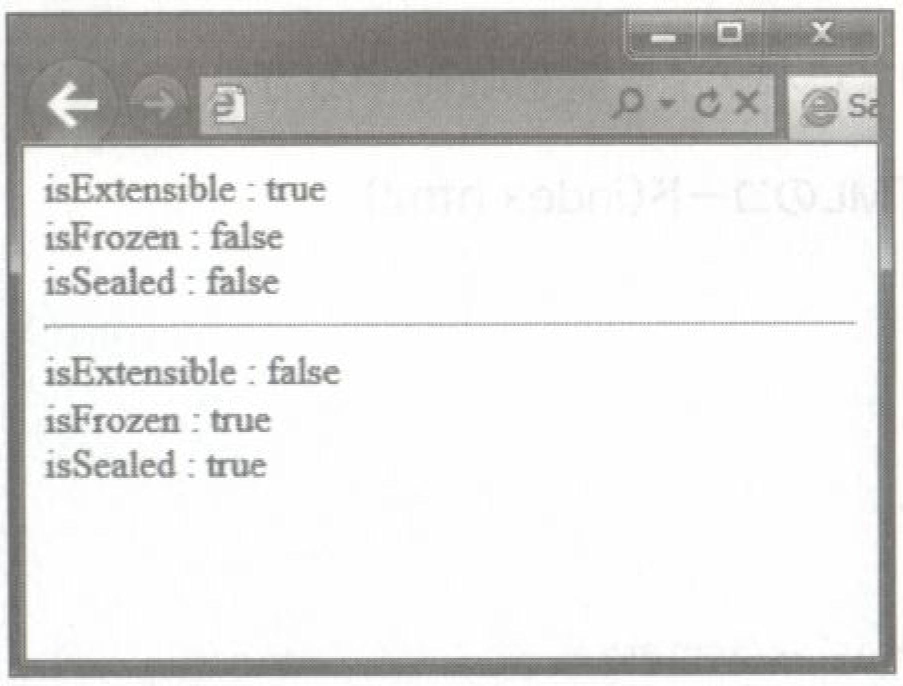
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オブジェクトを作成
  var myObj = {
    name : "MAPPY"
  };
  // オブジェクトの状態を表示
  ele.innerHTML = "isExtensible : "+Object.isExtensible(myObj)+"<br>";
  ele.innerHTML += "isFrozen : "+Object.isFrozen(myObj)+"<br>";
  ele.innerHTML += "isSealed : "+Object.isSealed(myObj)+"<hr>";
  // オブジェクトをロック
  Object.freeze(myObj);
  // オブジェクトの状態を表示
  ele.innerHTML += "isExtensible : "+Object.isExtensible(myObj)+"<br>";
  ele.innerHTML += "isFrozen : "+Object.isFrozen(myObj)+"<br>";
  ele.innerHTML += "isSealed : "+Object.isSealed(myObj);
}, false);
```

ブラウザでindex.htmlを表示すると、「freeze()」メソッドによってロックする前はオブジェクトの拡張などが可能になっています。「freeze()」メソッドの実行後(区切り線より下)はオブジェクトの変更が禁止されるため、拡張ができないことを示す結果になっています。



ONEPOINT

オブジェクトが拡張可能か確認するには「isExtensible」プロパティを使う

多くのブラウザに搭載されている新しいJavaScriptでは、不用意にオブジェクトの拡張を禁止する機能があります。オブジェクトが拡張可能かどうかは、「isExtensible」プロパティで確認することができます。「true」なら拡張可能で、「false」なら拡張不可です。

同様に、オブジェクトにプロパティを追加・削除、および、値の変更が可能かどうかは、「isFrozen」プロパティで確認することができます。「true」なら追加、および、値の変更が可能、「false」なら不可です。また、プロパティの追加・削除が可能かどうかは、「isSealed」プロパティで確認することができます。「true」なら追加が可能、「false」なら不可です。

なお、オブジェクトの拡張が禁止されている場合にプロパティを追加しても、例外エラーなどは発生しません。

関連項目 ▶▶▶

● オブジェクトの変更を禁止する …… p.104
● プロパティの追加を禁止する …… p.106

オブジェクトのプロパティ一覧を取得する

ここでは、オブジェクトのプロパティ一覧を取得する方法について解説します。

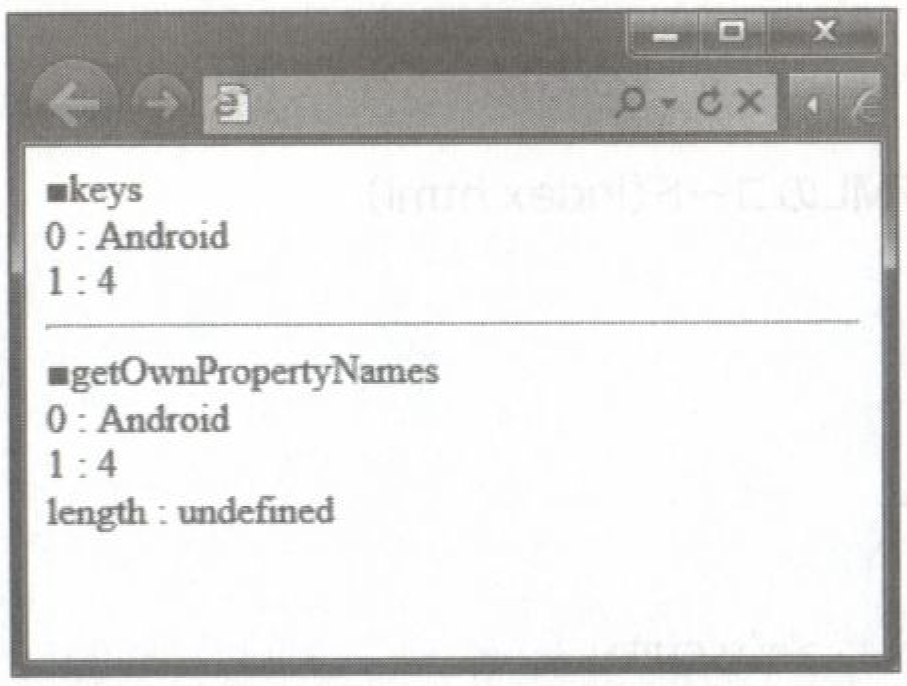
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列オブジェクトを作成
  var myObj = [ "Android", 4 ];
  // オブジェクトの内容をkeysで表示
  ele.innerHTML = "■keys<br>";
  var list1 = Object.keys(myObj);
  for(i=0; i<list1.length; i++){
    ele.innerHTML += list1[i]+" : " +myObj[i]+"<br>";
  }
  // オブジェクトの内容をgetOwnPropertyNamesで表示
  ele.innerHTML += "<hr>■getOwnPropertyNames<br>";
  var list2 = Object.getOwnPropertyNames(myObj);
  for(i=0; i<list2.length; i++){
    ele.innerHTML += list2[i]+" : " +myObj[i]+"<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、オブジェクトのプロパティ一覧が表示されます。「keys()」メソッドでは列挙型になっているプロパティは表示されませんが、「getOwnPropertyNames()」では列挙型を含むすべてのプロパティが表示されます。



ONEPOINT　オブジェクトのプロパティ一覧を取得するには「keys()」メソッドと「getOwnPropertyNames()」メソッドを使う

JavaScriptでオブジェクトのプロパティ一覧を取得する場合には、「for...in」文が使われてきました。現在のブラウザでは、これに加えて「keys()」メソッドと「getOwnPropertyNames()」メソッドを使ってプロパティを取得することができます。これらのメソッドは、パラメータで指定されたオブジェクトのプロパティ一覧を配列として返します。

「keys()」メソッドでは、列挙型となっているプロパティは返されません。「getOwnPropertyNames()」メソッドでは、列挙型も含めたプロパティを返します。

関連項目 ▶▶▶

- オブジェクトの内容を読み出す ………………………… p.96
- オブジェクトのプロパティの読み出し・書き込み処理を設定する ………… p.102

SECTION-033

オブジェクトの種類を調べる

ここでは、オブジェクトの種類を調べる方法について解説します。

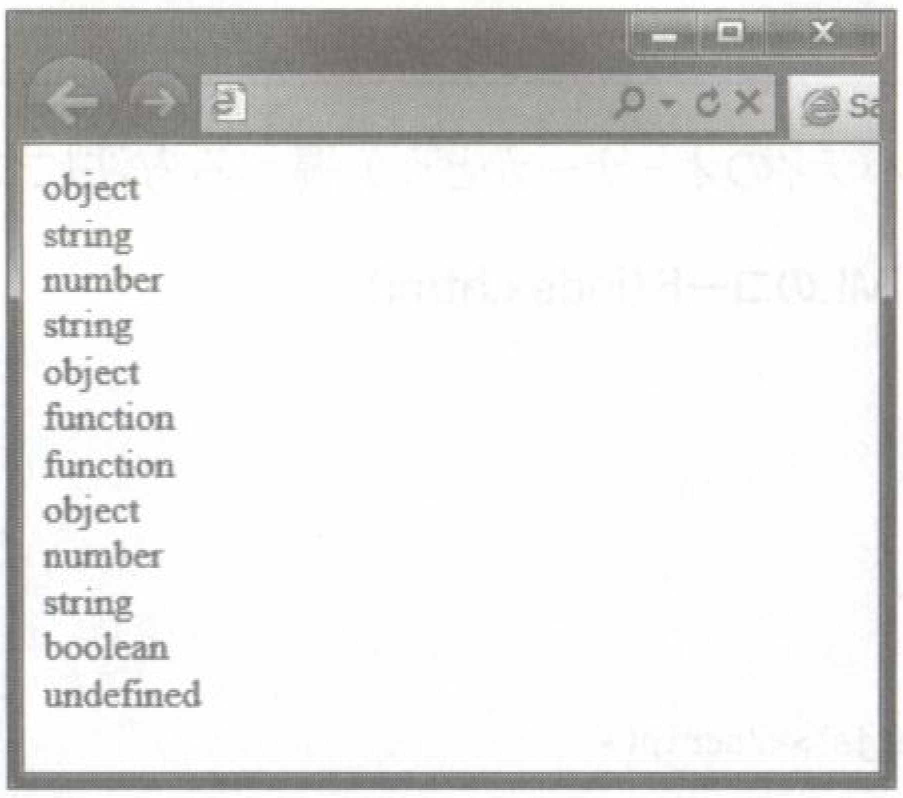
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列に各種オブジェクトを用意
  var myObj = [
    Array(1), String(2), Number(3), Date(), Math, RegExp,
    function(){}, null, 0.123, "OK", false, undefined
  ];
  // オブジェクトの種類を表示
  for(var i=0; i<myObj.length; i++){
    // 種類を調べる
    var type = typeof(myObj[i]);
    ele.innerHTML += type+"<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、各種オブジェクトを調べた結果が表示されます。なお、「typeof」演算子では、どのオブジェクトのインスタンスなのかは調べることはできません。



ONEPOINT　オブジェクトの種類を調べるには「typeof」演算子を使う

JavaScriptでオブジェクトの種類を調べるには、「typeof」演算子を使います。パラメータに調べたいオブジェクトや変数などを指定します。オブジェクトの種類に応じて、次の表の文字列が返されます。なお、「typeof」演算子は種類を返すだけであり、インスタンスがどのオブジェクトから生成されたものかを把握することはできません。どのオブジェクトのインスタンス化を調べる場合は、「instanceOf」演算子を使う必要があります。

文字列	説明
boolean	真偽(ブーリアン)
object	各種オブジェクト(「null」も該当)
number	数値
string	文字
function	関数(正規表現も該当)
undefined	未定義

SECTION-034

プロパティのオーナーかどうか調べる

ここでは、指定したプロパティのオーナーかどうか調べる方法について解説します。

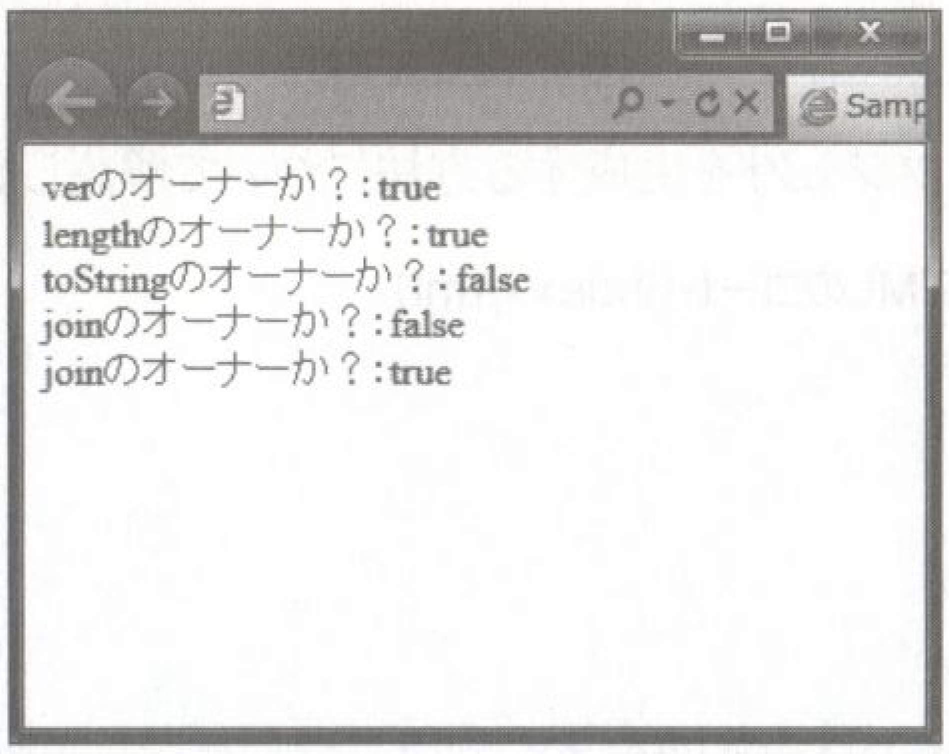
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列オブジェクトを生成
  var a = new Array(1,2,3,4);
  // 新たなプロパティを追加
  a.ver = 2.5;
  // 追加した新たなプロパティであるverの所有者か調べる
  ele.innerHTML = "verのオーナーか？ :"+a.hasOwnProperty("ver")+"<br>";
  // lengthの所有者か調べる
  ele.innerHTML += "lengthのオーナーか？ :"+a.hasOwnProperty("length")+"<br>";
  // toStringの所有者か調べる
  ele.innerHTML += "toStringのオーナーか？ :"+a.hasOwnProperty("toString")+"<br>";
  // joinの所有者か調べる
  ele.innerHTML += "joinのオーナーか？ :"+a.hasOwnProperty("join")+"<br>";
  // inを使ってjoinの所有者か調べる
  ele.innerHTML += "joinのオーナーか？ :"+("join" in a)+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、配列オブジェクトのプロパティについて、オーナーかどうか調べた結果が表示されます。



ONEPOINT

プロパティのオーナーかどうか調べるには「hasOwnProperty()」メソッドを使う

プロパティのオーナーかどうか調べるには、「hasOwnProperty()」メソッドを使います。「hasOwnProperty()」は各オブジェクトのメソッドとして用意されており、パラメータに調べたいプロパティ名を文字列で指定します。

「hasOwnProperty()」メソッドは、オブジェクトのプロトタイプチェーンをたどって調べることはしません。このため、純粋にオブジェクトがそのプロパティを持っているかどうか調べることができます。オブジェクトがプロパティを持っている場合は「true」、そうでない場合は「false」を返します。プロトタイプチェーンをたどって調べる場合には「hasOwnProperty()」メソッドではなく、「in」演算子を使います。

関連項目 ▶▶▶

- オブジェクトの内容を読み出す ………… p.96
- プロトタイプを利用する ………… p.98
- プロパティの属性値を設定・取得する ………… p.100

「Boolean」オブジェクトを生成する

ここでは、「Boolean」オブジェクトを生成する方法について解説します。

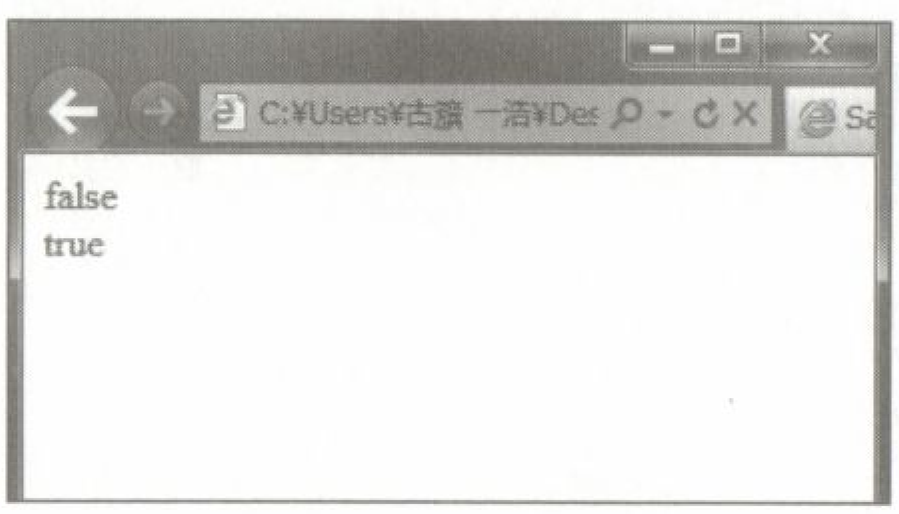
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数flag1をBoolean(真偽)オブジェクトとして生成
  var flag1 = new Boolean();
  // 変数flag2をBoolean(真偽)オブジェクトで値をtrueにして生成
  var flag2 = new Boolean(true);
  // 結果を出力
  ele.innerHTML = flag1+"<br>"+flag2;
}, false);
```

ブラウザでindex.htmlを表示すると、「Boolean」オブジェクトで生成された変数の結果が表示されます。



ONEPOINT 「Boolean」オブジェクトを生成するには「new Boolean()」とする

JavaScriptで真偽値を持つオブジェクト(「Boolean」オブジェクト)を生成するには、「new Boolean()」とします。パラメータに何も指定しない場合や内容が偽(「false」)の場合、変数は「false」になります。パラメータが真(「true」)の場合は「true」になります。

SECTION-036

プロパティやオブジェクトを削除する

ここでは、プロパティやオブジェクトを削除する方法について説明します。

SAMPLE CODE HTMLのコード(index.html)

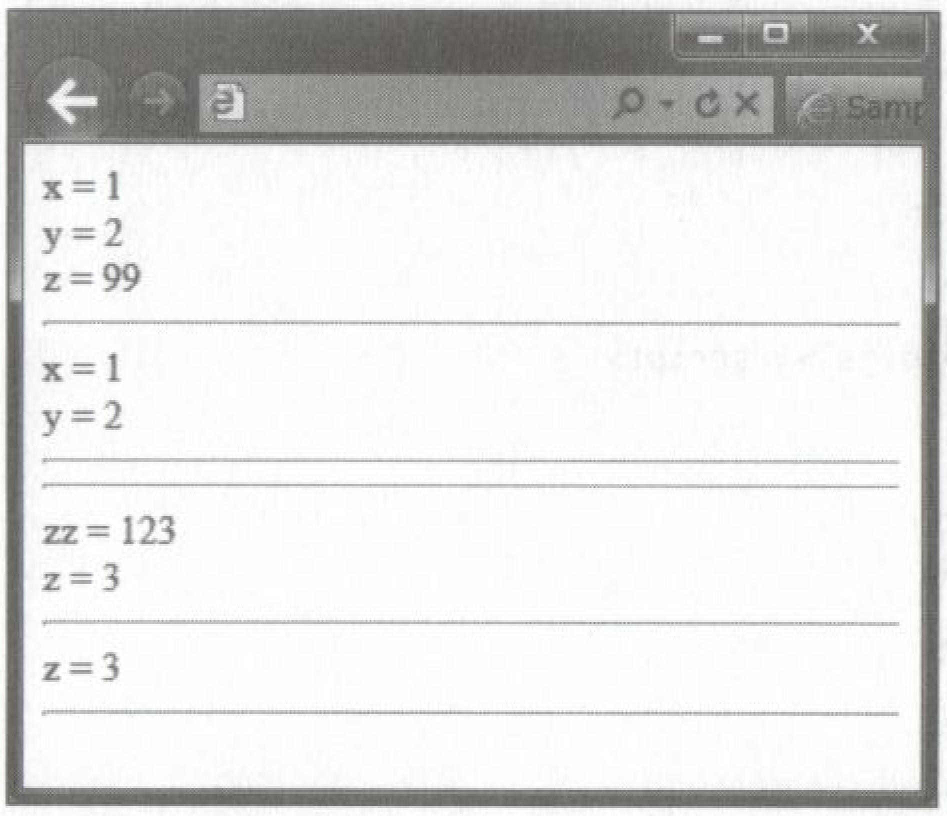
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output><hr>
    <output></output><hr>
    <output></output><hr>
    <output></output><hr>
    <output></output><hr>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // オブジェクトを作成
  var myObj = { x : 1, y : 2, z : 99 };
  // 内容を表示
  for(var i in myObj){ ele[0].innerHTML += i + " = " + myObj[i]+"<br>"; }
  // zプロパティだけ削除
  delete myObj.z;
  // 内容を表示
  for(var i in myObj){ ele[1].innerHTML += i + " = " + myObj[i]+"<br>"; }
  // オブジェクトを削除
  myObj = null;
  // 内容を表示。オブジェクトがないので内も出力されない
  for(var i in myObj){ ele[2].innerHTML += i + " = " + myObj[i]+"<br>"; }
  // オブジェクトを新たに作成
  var tempObj = function(){ this.zz = 123; };
  tempObj.prototype.z = 3;
  var myObj = new tempObj();
  // 内容を表示
  for(var i in myObj){ ele[3].innerHTML += i + " = " + myObj[i]+"<br>"; }
  // z, zzプロパティを削除。自分自身のプロパティであるzzしか削除できない
  delete myObj.z;
```

```
    delete myObj.zz;
    // 内容を表示
    for(var i in myObj){ ele[4].innerHTML += i + " = " + myObj[i]+"<br>"; }
}, false);
```

ブラウザでindex.htmlを表示すると、作成したオブジェクトの内容と、削除後の内容が表示されます。削除されたオブジェクトの内容は表示されません。



ONEPOINT オブジェクトを削除するには「null」を代入、プロパティを削除するには「delete」演算子を使う

オブジェクトを削除するには、「null」を代入します。プロパティを削除するには、「delete」演算子を使います。ただし、「delete」演算子でどんなプロパティでも削除できるわけではありません。自分自身が持つプロパティのみ削除でき、プロトタイプチェーンをたどって返されたプロパティは削除されません。また、「delete」演算子では「var」宣言した変数を削除することはできません。

関連項目 ▶▶▶

- オブジェクトを生成する …… p.90
- 指定されたオブジェクトを継承して新たなオブジェクトを生成する …… p.92
- プロパティの属性値を設定・取得する …… p.100
- オブジェクトの変更を禁止する …… p.104
- オブジェクトの変更・拡張などが可能か調べる …… p.108

CHAPTER 03

配列

SECTION-037

配列オブジェクトを生成する

ここでは、配列オブジェクトを生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = new Array();
  // 配列要素を10個作成
  var myArray2 = new Array(10);
  // 配列要素をあらかじめ代入して作成
  var myArray3 = new Array("C","and","R");
  // 配列要素をあらかじめ代入して作成
  var myArray3 = [10];
  // 二次元配列を作成
  var myArray4 = [[1,2],[3,4,5],[6,7,8,9]];
  // 配列の内容を出力
  ele.innerHTML = "■配列内容";
  ele.innerHTML += "<br>myArray1 = "+myArray1.toString();
  ele.innerHTML += "<br>myArray2 = "+myArray2.toString();
  ele.innerHTML += "<br>myArray3 = "+myArray3.toString();
  ele.innerHTML += "<br>myArray4 = "+myArray4.toString();
  ele.innerHTML += "<hr>■配列の長さ";
  ele.innerHTML += "<br>myArray1 = "+myArray1.length;
  ele.innerHTML += "<br>myArray2 = "+myArray2.length;
  ele.innerHTML += "<br>myArray3 = "+myArray3.length;
  ele.innerHTML += "<br>myArray4 = "+myArray4.length;
}, false);
```

ブラウザでindex.htmlを表示すると、生成した配列の内容と配列の長さ(要素総数)が表示されます。



ONEPOINT 配列を生成するには「new Array()」や「[]」を使う

JavaScriptで配列を生成するには、「new Array()」として配列オブジェクトを生成します。何もパラメータを指定しない場合は、空っぽの配列が生成されます。JavaScriptでは配列要素は自動的に調整されるため、あらかじめ要素数を指定する必要はありません。それでも明示的に要素数を指定する場合は、「new Array(10)」のように数値のパラメータを1つだけ指定します。この場合、配列要素が10個、用意されます。パラメータが複数ある場合は、それらは要素として代入されます。たとえば、「new Array(10, 20)」とした場合、最初の配列要素に「10」の数値、その次の要素に「20」の数値が入ります。

また、「new Array()」とする代わりに、「[]」と記述することもできます。ここで、「a = [10]」とした場合は、「a = new Array(10)」とは動作が異なる点には注意が必要です。「[]」を使った場合、指定した内容の数にかかわらず、すべて配列の要素として代入されます。

配列は一次元配列だけでなく、入れ子にすることで二次元配列など、多次元配列を生成することができます。

なお、新しいブラウザでは配列要素のビット数を固定して生成することができます。これに関しては、122ページを参照してください。

関連項目 ▶▶▶

● 指定したビット長の配列を生成する ………………………………………… p.122

SECTION-038

指定したビット長の配列を生成する

ここでは、指定したビット長の配列を生成する方法について解説します。

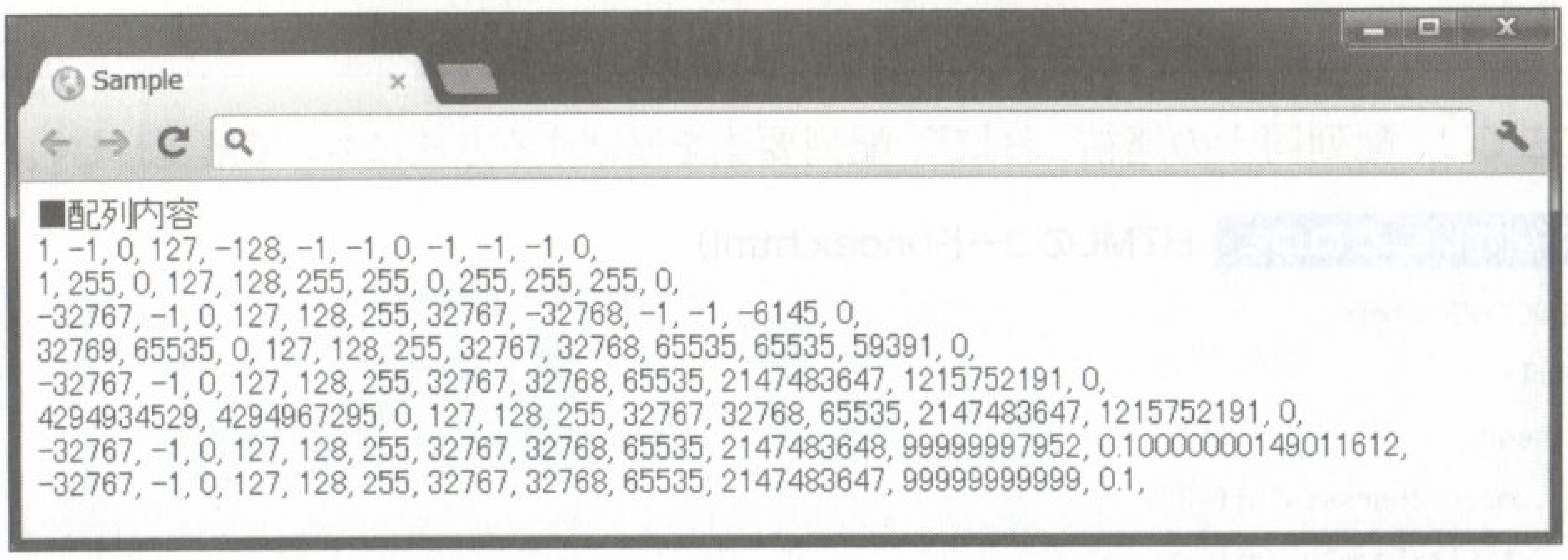
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // データを入れる配列を用意
  myArray = new Array();
  myArray[0] = new Int8Array(12);     // 符号付き8ビット長配列を作成
  myArray[1] = new Uint8Array(12);    // 符号なし8ビット長配列を作成
  myArray[2] = new Int16Array(12);    // 符号付き16ビット長配列を作成
  myArray[3] = new Uint16Array(12);   // 符号なし16ビット長配列を作成
  myArray[4] = new Int32Array(12);    // 符号付き32ビット長配列を作成
  myArray[5] = new Uint32Array(12);   // 符号なし32ビット長配列を作成
  myArray[6] = new Float32Array(12);  // 浮動小数32ビット長配列を作成
  myArray[7] = new Float64Array(12);  // 浮動小数64ビット長配列を作成
  // 値を代入し結果を表示
  ele.innerHTML = "■配列内容<br>";
  var n = [-32767, -1, 0, 127, 128, 255, 32767, 32768, 65535, 2147483647, 9999999999, 0.1];
  for(var j=0; j<myArray.length; j++){
    for(var i=0; i<n.length; i++){
      myArray[j][i] = n[i];
      ele.innerHTML += myArray[j][i] + ", ";
    }
    ele.innerHTML += "<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、生成した配列の内容が表示されます。ビット長が制限されているため、同じ値を代入しても異なる結果になります。



ONEPOINT ビット長を指定して配列要素を生成するには「Typed Array」を使う

JavaScriptでは、配列要素にさまざまなデータを代入することができます。数字や文字列だけでなく、オブジェクトも代入することができます。しかし、用途によっては配列要素のビット長が決まっている方が便利な場合があります。たとえば、画像処理やオーディオの波形処理などです。JavaScriptでは「Typed Array」といってビット長さを固定して配列を生成することができます。その際、使用する要素数を指定する必要があります。ビット長を指定して生成できる配列は、次の表に示すものになります。

オブジェクト(コンストラクタ)	説明
Int8Array	符号付き8ビットデータ
Uint8Array	符号なし8ビットデータ
Int16Array	符号付き16ビットデータ
Uint16Array	符号なし16ビットデータ
Int32Array	符号付き32ビットデータ
Uint32Array	符号なし32ビットデータ
Float32Array	浮動小数32ビット(IEEE754)
Float64Array	浮動小数32ビット(IEEE754)

また、「ArrayBuffer」オブジェクトを使ってバイト単位で同一の配列にアクセスすることもできます。次の例では、「byteData2[0]」には「255」、「byteData2[1]」には「127」が入ります。

```
// データを入れる配列を16バイト分用意
var buffer = new ArrayBuffer(16);
byteData1 = new Uint16Array(buffer);
byteData2 = new Uint8Array(buffer);
byteData1[0] = 32767;
```

関連項目 ▶▶▶

●配列オブジェクトを生成する ……………………………… p.120

配列同士や配列要素を連結する

ここでは、配列同士の連結、および、配列要素を連結する方法について解説します。

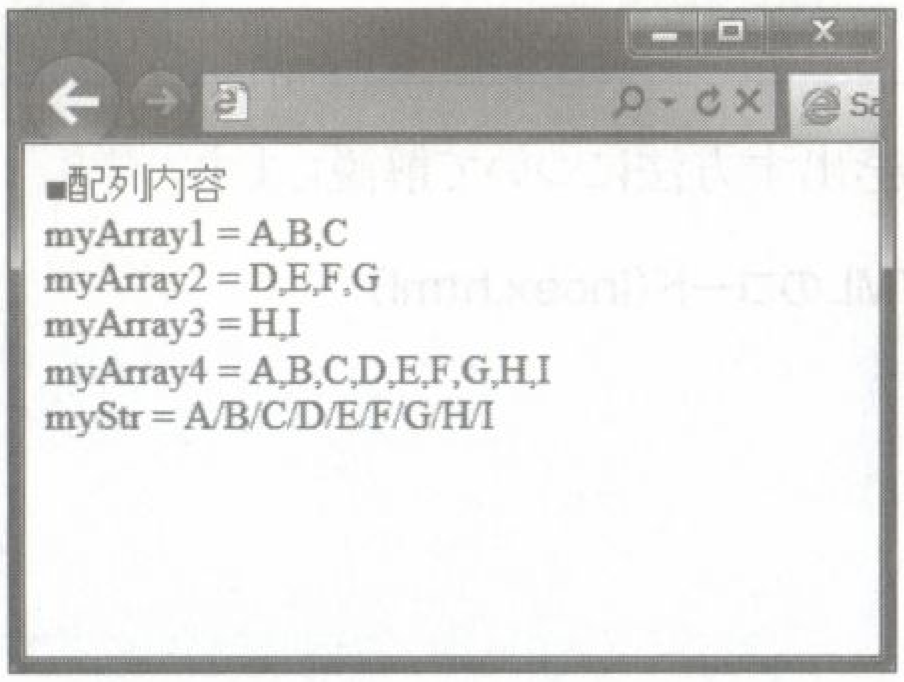
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = ["A", "B", "C"];
  var myArray2 = ["D", "E", "F", "G"];
  var myArray3 = ["H", "I"];
  // 配列同士を連結
  var myArray4 = myArray1.concat(myArray2, myArray3);
  // 配列要素をあらかじめ代入して作成
  var myStr = myArray4.join("/");
  // 配列の内容を出力
  ele.innerHTML = "■配列内容";
  ele.innerHTML += "<br>myArray1 = "+myArray1.toString();
  ele.innerHTML += "<br>myArray2 = "+myArray2.toString();
  ele.innerHTML += "<br>myArray3 = "+myArray3.toString();
  ele.innerHTML += "<br>myArray4 = "+myArray4.toString();
  ele.innerHTML += "<br>myStr = "+myStr;
}, false);
```

ブラウザでindex.htmlを表示すると、連結する前の配列内容と配列同士を連結した結果、および、配列の各要素を特定の文字で連結した結果が表示されます。



ONEPOINT **配列同士を連結するには「concat()」メソッド、配列要素を連結するには「join()」メソッドを使う**

配列同士を連結するには、「concat()」メソッドを使います。「concat()」メソッドは配列のメソッドとして用意されており、パラメータに連結したい配列を指定します。その際、複数の配列を指定すると、指定した順番に配列が連結されていきます。

配列の要素を特定の区切り文字で連結して文字列にしたい場合には、「join()」メソッドを使います。配列の各要素が「join()」メソッドのパラメータに指定した文字で連結されます。「join()」メソッドのパラメータを省略することができ、その場合は「,」(カンマ)で各要素が連結され文字列として返されます。「concat()」メソッド、「join()」メソッドとも、配列内容は変化しません(配列は非破壊)。

COLUMN **IE6で高速に文字列を連結する**

すでにBADノウハウですが、IE6などの古いIEでは、「+」記号を使った文字列の連結が他のブラウザと比べて格段に低速です。これを回避する方法の1つとして配列に文字列を入れておき、最後に「join()」メソッドを使って連結するという方法があります。

IE9などの新しいバージョンでは、文字列連結などは全般的に処理が高速化されており、このノウハウは役に立ちません。

関連項目 ▶▶▶

● 「String」オブジェクトを生成する/文字列を連結する …… p.194

SECTION-040

配列要素を抜き出す

ここでは、配列要素を抜き出す方法について解説します。

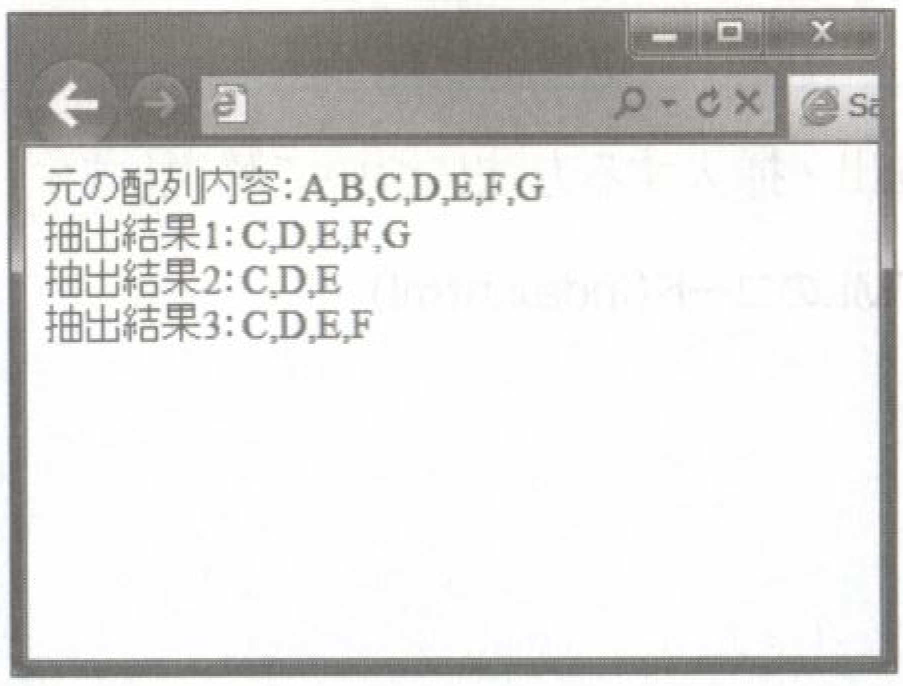
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["A", "B", "C", "D", "E", "F", "G"];
  // 配列から3番目以降の要素を抜き出す
  var data1 = myArray.slice(2);
  // 配列から3～5番目の要素を抜き出す
  var data2 = myArray.slice(2, 5);
  // 配列から3番目～最後から2番目の要素を抜き出す
  var data3 = myArray.slice(2, -1);
  // 配列の内容を出力
  ele.innerHTML = "元の配列内容:"+myArray.toString()+"<br>";
  ele.innerHTML += "抽出結果1:"+data1.toString()+"<br>";
  ele.innerHTML += "抽出結果2:"+data2.toString()+"<br>";
  ele.innerHTML += "抽出結果3:"+data3.toString()+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、元となる配列から要素を抜き出した結果が表示されます。



ONEPOINT 配列要素を抜き出すには「slice()」メソッドを使う

配列の特定範囲の要素を抜き出すには、「slice()」メソッドを使います。「slice()」メソッドは、指定するパラメータの数と値によって異なる動作をします。「slice()」メソッドには、1つ、または、2つのパラメータを指定します。パラメータは、配列の要素を示す数値になります。「0」が配列の先頭要素を示し、「-2」であれば最後から2番目の要素を示します。

パラメータが1つの場合は、指定した要素から最後の要素まで抜き出されます。パラメータが2つの場合は、指定した範囲の要素が抜き出されます。「slice()」メソッドを実行すると、抜き出した結果は配列で返されます(要素は参照でなくコピーされる)。その際、元の配列要素は何も変化しません(配列は非破壊)。

関連項目

●配列要素を抽出・挿入する……p.128

配列要素を抽出・挿入する

ここでは、配列要素を抽出・挿入する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

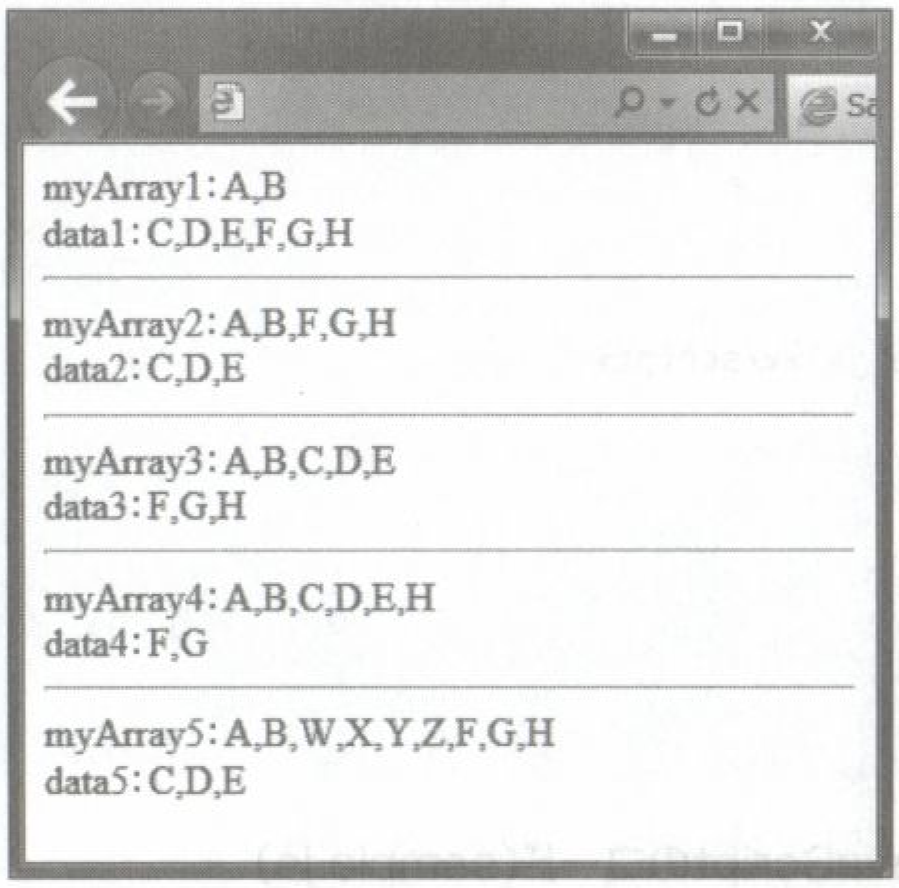
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = ["A", "B", "C", "D", "E", "F", "G", "H"];
  var myArray2 = ["A", "B", "C", "D", "E", "F", "G", "H"];
  var myArray3 = ["A", "B", "C", "D", "E", "F", "G", "H"];
  var myArray4 = ["A", "B", "C", "D", "E", "F", "G", "H"];
  var myArray5 = ["A", "B", "C", "D", "E", "F", "G", "H"];
  // 3番目以降の要素を抜き出す
  var data1 = myArray1.splice(2);
  // 3番目から3つの要素を抜き出す
  var data2 = myArray2.splice(2, 3);
  // 最後から3つの要素を抜き出す
  var data3 = myArray3.splice(-3);
  // 最後から2番目と3番目の要素を抜き出す(抜き出す位置が最後から3番目で、そこから2つ抜き出す)
  var data4 = myArray4.splice(-3, 2);
  // 3番目から3つの要素を抜き出し、3番目以降に新たな要素を挿入
  var data5 = myArray5.splice(2, 3, "W", "X", "Y", "Z");
  // 配列の内容を出力
  ele.innerHTML = "myArray1:"+myArray1.toString()+"<br>";
  ele.innerHTML += "data1:"+data1.toString()+"<hr>";
  ele.innerHTML += "myArray2:"+myArray2.toString()+"<br>";
  ele.innerHTML += "data2:"+data2.toString()+"<hr>";
  ele.innerHTML += "myArray3:"+myArray3.toString()+"<br>";
  ele.innerHTML += "data3:"+data3.toString()+"<hr>";
  ele.innerHTML += "myArray4:"+myArray4.toString()+"<br>";
```

```
    ele.innerHTML += "data4:"+data4.toString()+"<hr>";
    ele.innerHTML += "myArray5:"+myArray5.toString()+"<br>";
    ele.innerHTML += "data5:"+data5.toString();
}, false);
```

ブラウザでindex.htmlを表示すると、変更された元の配列の内容と抽出・挿入した結果の配列が表示されます。



ONEPOINT 配列要素を抽出・挿入するには「splice()」メソッドを使う

配列要素を抽出・挿入するには、「splice()」メソッドを使います。「splice()」メソッドは、元の配列から取り除いた結果を返します(結果は配列で返される)。また、元の配列内容は変更されます(配列は破壊)。ただし、「splice()」メソッドは、パラメータの数などによって動作が変わります。

まず、パラメータが1つの場合は、指定した位置にある要素から最後の要素までを元の配列から取り除きます。また、パラメータが負数の場合は、最後の要素から数えた位置になります。たとえば、「splice(-2)」であれば、最後から2つの要素が取り除かれます。

パラメータが2つの場合は、要素を抜き出す範囲になります。最初のパラメータが抜き出す位置になります(負数も指定可能)。2番目のパラメータは、抜き出す要素数になります。

パラメータが3つ以上の場合は、要素を抜き出した位置に新たな要素が挿入されます。これは「splice()」メソッドの3番目以降で指定された要素が順番に元の配列に挿入されます。

関連項目 ▶▶▶

● 配列要素を抜き出す ………… p.126

条件を満たす配列要素を調べる

ここでは、条件を満たす配列要素を調べる方法について解説します。

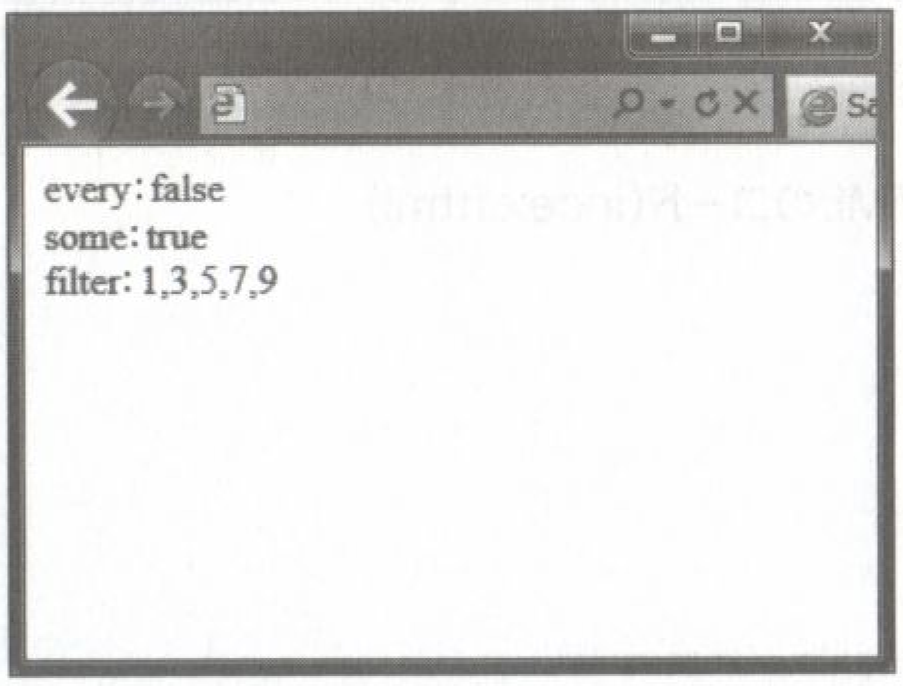
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = [1, 2, 3, 4, 5, 6, 7, 8, 9];
  // 配列要素が5未満かどうか調べる
  var flag1 = myArray.every(check);
  var flag2 = myArray.some(check);
  // 配列から偶数を取り除く
  var ary = myArray.filter(function(element, index, targetArray){
    if (element & 1){
      return element;
    }
  });
  // 判断するコールバック関数
  function check(element, index, targetArray){
    // 読み出した要素を調べる
    if (element < 5){
      return true;
    }else{
      return false;
    }
  }
  // 結果を出力
  ele.innerHTML = "every:"+flag1+"<br>";
  ele.innerHTML += "some:"+flag2+"<br>";
  ele.innerHTML += "filter:"+ary.toString();
}, false);
```

ブラウザでindex.htmlを表示すると、「every()」メソッドと「some()」メソッドを使って、配列要素を調べた結果が表示されます。その後、「filter()」メソッドで奇数だけを抽出した結果が表示されます。



ONEPOINT

配列要素が条件を満たしているか調べるには「every()」メソッドや「some()」メソッドを使う

配列要素が条件を満たしているかどうかは、「every()」メソッドか「some()」メソッドを使います。どちらのメソッドもパラメータにはコールバック関数を指定します。コールバック関数には、パラメータとして、配列要素、配列要素の位置、対象となる配列が渡されます。

「every()」はコールバック関数からの戻り値がすべて「true」の場合、「true」を返します。コールバック関数からの戻り値が「false」の場合は、その時点で「false」を返します。

「some()」はコールバック関数からの戻り値がすべて「false」の場合、「false」を返します。コールバック関数からの戻り値が「true」の場合は、その時点で「true」を返します。

また、条件を満たした配列要素だけを抽出したい場合には、「filter()」メソッドを使います。パラメータには、コールバック関数を指定します。コールバック関数に渡されるパラメータは、「every()」メソッドや「some()」メソッドと同じです。「filter()」メソッドは、コールバック関数内から「return」文によって返された要素だけを新たな配列の要素とします。なお、「filter()」メソッドを実行しても元の配列は変化しません（非破壊）。

SECTION-043

配列要素を順番に処理する

ここでは、配列要素を順番に処理する方法について解説します。

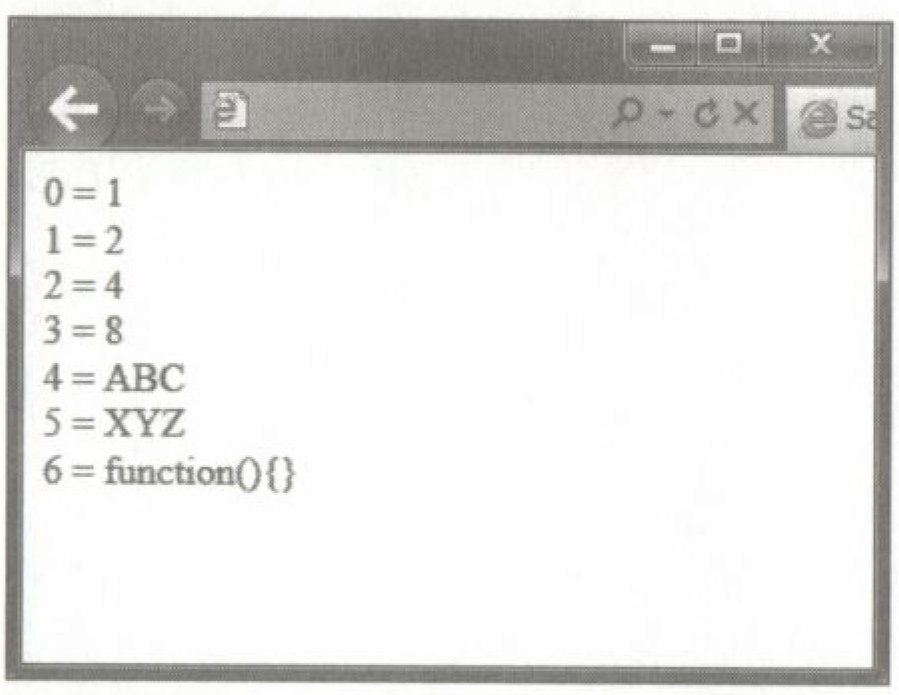
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = [1, 2, 4, 8, "ABC", "XYZ", function(){}];
  // 配列を順番に処理
  myArray.forEach(function(element, index, targetArray){
    // 読み出した要素の順番と内容を表示
    ele.innerHTML += index+" = "+element+"<br>";
  });
}, false);
```

ブラウザでindex.htmlを表示すると、すべての配列要素のインデックス番号と内容が表示されます。



ONEPOINT　配列要素を順番に処理するには「forEach()」メソッドを使う

配列要素を順番に処理する場合には、「forEach()」メソッドを使います。「for」文を使って配列を処理することもできますが、「forEach()」メソッドの場合は繰り返しのための変数を用意する必要がありません。「forEach()」のようなメソッドは、イテレーターとも呼ばれます。また、「forEach()」メソッドのパラメータには、配列要素の処理を行う関数を指定します。この関数には「要素内容」「参照番号」「対象となっている配列オブジェクト」が渡されます。

「forEach()」メソッドには、2番目のパラメータも指定することができます。2番目のパラメータには「forEach()」メソッドで呼び出される関数内で示されるthisとなるオブジェクトを指定します。

なお、古いブラウザでは「forEach()」メソッドは実装されていませんが、次のURLにあるコードを使うことで同様の機能を実装することができます。

URL https://developer.mozilla.org/ja/Core_JavaScript_1.5_Reference/Global_Objects/Array/forEach

COLUMN　重複している要素を排除するには

配列の名前や商品名などのデータが重複して入っている場合、次のようにハッシュ(連想配列)を利用することで重複している項目を排除することができます。次の例では、「PC」「MZ」「FM」の3つが重複していますが、プログラムを実行すると、「PC」「MZ」「FM」の3つだけにまとめられ表示されます。

```
var ele = document.getElementsByTagName("output")[0];
// 重複している項目がある配列を作成
var myArray = ["PC", "MZ", "FM", "PC", "PC", "MZ", "FM", "MZ", "MZ"];
// 結果を入れる配列を作成
var resultArray = [ ];
// 要素の数だけ繰り返す
for(var i=0; i<myArray.length; i++){
  resultArray[myArray[i]] = true;
}
// 結果を出力
for(var i in resultArray){
  ele.innerHTML += i+"<br>";
}
```

関連項目▶▶▶

- 繰り返し処理(ループ)について ………… p.54
- 配列要素を加工する ………… p.140
- 配列要素を前方または後方から処理する ………… p.144

SECTION-044

配列要素を検索する

ここでは、配列要素を検索する方法について解説します。

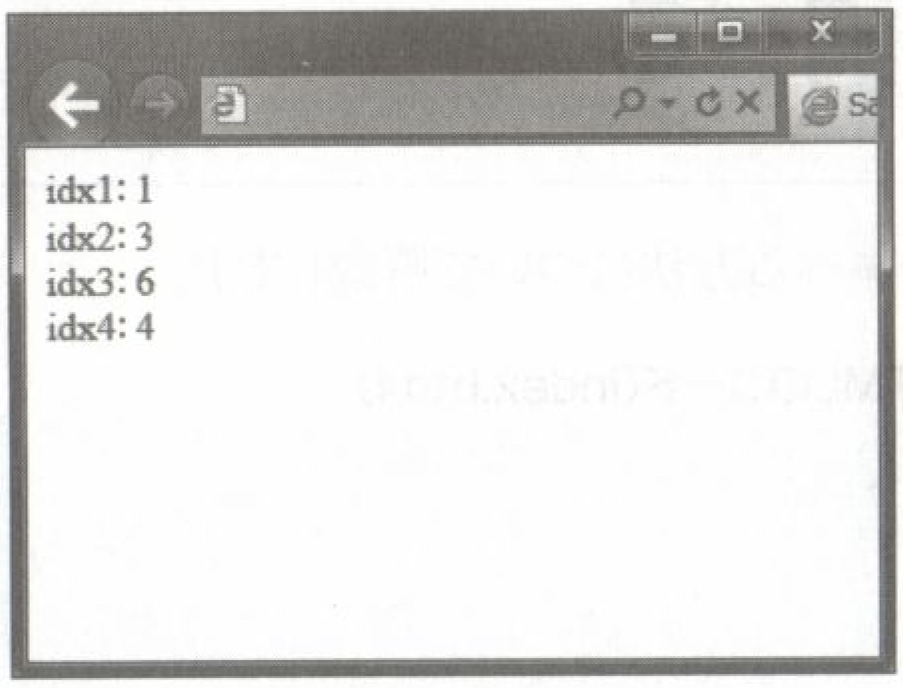
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["Basic", "ルビ", "ADA", "ルビ", "ルビ", "Basic", "ルビ", "Perl"];
  // 配列を先頭から後方に検索
  var idx1 = myArray.indexOf("ルビ");
  // 配列を3番目の要素から後方に検索
  var idx2 = myArray.indexOf("ルビ", 2);
  // 配列を最後から前方に検索
  var idx3 = myArray.lastIndexOf("ルビ");
  // 配列を最後の3番目から前方に検索
  var idx4 = myArray.lastIndexOf("ルビ", -3);
  // 結果を出力
  ele.innerHTML = "idx1:"+idx1+"<br>";
  ele.innerHTML += "idx2:"+idx2+"<br>";
  ele.innerHTML += "idx3:"+idx3+"<br>";
  ele.innerHTML += "idx4:"+idx4;
}, false);
```

ブラウザでindex.htmlを表示すると、配列要素を検索した結果が表示されます。



ONEPOINT

配列要素を検索するには「indexOf()」メソッドや「lastIndexOf()」メソッドを使う

配列要素を前方から検索するには、「indexOf()」メソッドを使います。これは、文字列の検索に使われる「indexOf()」メソッドと似ています。最初のパラメータに、要素の内容を指定します。該当する要素が見つかったら、その要素の位置を返します。要素が見つからなかった場合は、「-1」を返します。「indexOf()」メソッドの2番目のパラメータは、省略することができます。指定した場合は、指定した位置から要素の検索を行います。位置が配列要素よりも大きい場合には、「-1」を返します。位置に負数を指定した場合は、後方からの位置を示します。

配列要素を後方から検索する場合は、「lastIndexOf()」メソッドを使います。パラメータは「indexOf()」メソッドと同じです。

なお、古いブラウザでは「indexOf()」メソッドと「lastIndexOf()」メソッドは実装されていませんが、次のURLにあるコードを使うことで同等の機能を実装することができます。

URL https://developer.mozilla.org/ja/JavaScript/Reference/Global_Objects/Array/indexOf

URL https://developer.mozilla.org/ja/JavaScript/Reference/Global_Objects/Array/lastIndexOf

関連項目 ▶▶▶

● 文字列を検索する …… p.200

配列かどうか調べる

ここでは、配列かどうか調べる方法について解説します。

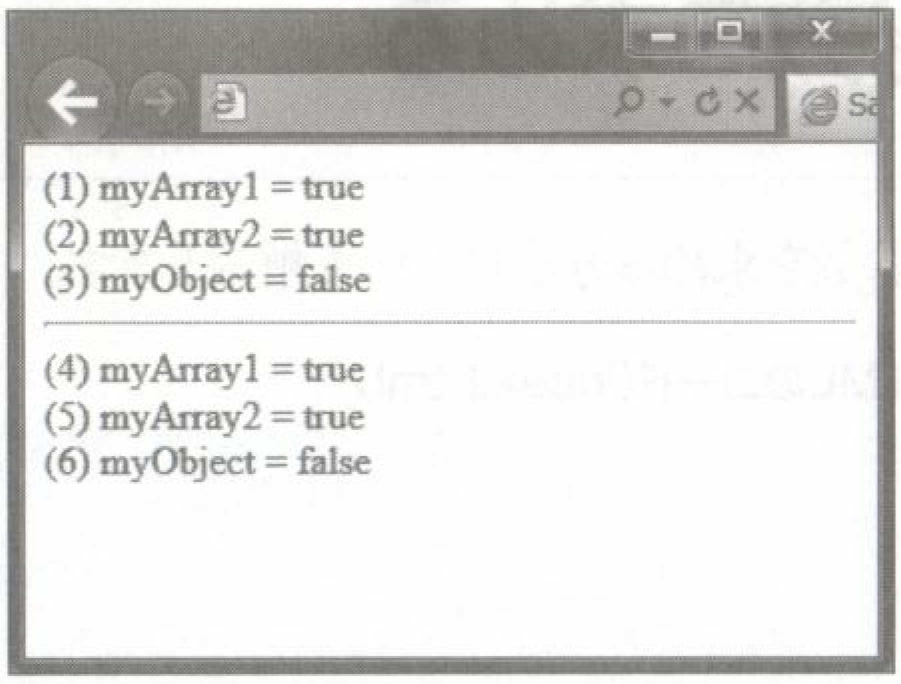
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = new Array();
  // 配列を作成
  var myArray2 = ["a-Ray"];
  // オブジェクトを作成
  var myObject = new Object();
  // 配列かどうか調べる
  var flag1 = Array.isArray(myArray1);
  var flag2 = Array.isArray(myArray2);
  var flag3 = Array.isArray(myObject);
  var flag4 = myArray1 instanceof Array;
  var flag5 = myArray2 instanceof Array;
  var flag6 = myObject instanceof Array;
  // 結果を出力
  ele.innerHTML = "(1) myArray1 = "+flag1+"<br>";
  ele.innerHTML += "(2) myArray2 = "+flag2+"<br>";
  ele.innerHTML += "(3) myObject = "+flag3+"<hr>";
  ele.innerHTML += "(4) myArray1 = "+flag4+"<br>";
  ele.innerHTML += "(5) myArray2 = "+flag5+"<br>";
  ele.innerHTML += "(6) myObject = "+flag6+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、配列かどうか調べた結果が表示されます。



ONEPOINT　**配列かどうか調べるには「instanceof」演算子や「Array.isArray()」メソッドを使う**

JavaScriptでは配列かどうか調べるには、「instanceof」演算子を使う方法と「Array.isArray()」メソッドを使う方法があります。「Array.isArray()」はIE9以降で使用できるので、それ以前のバージョンで動作させる場合は「instanceof」演算子を使います。どちらも配列の場合は「true」を、そうでない場合は「false」を返します。

なお、古いブラウザでは「isArray()」メソッドは実装されていませんが、次のURLにあるコードを使うことで同等の機能を実装することができます。

URL https://developer.mozilla.org/ja/JavaScript/Reference/Global_Objects/Array/isArray

関連項目 ▶▶▶

● 数値かどうか調べる …… p.80
● オブジェクトの種類を調べる …… p.112

配列要素の総数を求める

ここでは、配列要素の総数を求める方法について解説します。

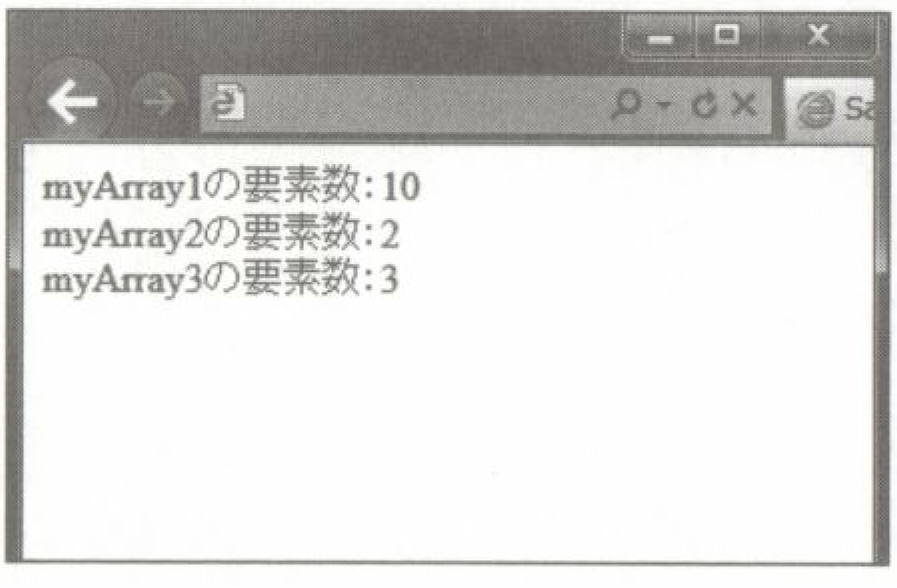
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 生成数を指定して配列を作成
  var myArray1 = new Array(10);
  // 配列を作成
  var myArray2 = new Array(10, 20);
  // 配列を作成
  var myArray3 = ["H", "He", "Li"];
  // 配列の数を求める
  var n1 = myArray1.length;
  var n2 = myArray2.length;
  var n3 = myArray3.length;
  // 結果を出力
  ele.innerHTML = "myArray1の要素数 :"+n1+"<br>";
  ele.innerHTML += "myArray2の要素数 :"+n2+"<br>";
  ele.innerHTML += "myArray3の要素数 :"+n3;
}, false);
```

ブラウザでindex.htmlを表示すると、それぞれの配列要素の総数が表示されます。



ONEPOINT

配列の要素数を調べるには「length」プロパティを使う

JavaScriptでは配列の総数を調べるには、「length」プロパティを参照します。配列の「length」プロパティは読み出しだけでなく、次のようにサイズを指定して設定することもできます。

```
var ele = document.getElementsByTagName("output")[0];
// 空っぽの配列を作成
var myArray = [ ];
// 配列の数を設定する
myArray.length = 10;
// 結果を出力
ele.innerHTML = "myArrayの要素数 :"+myArray.length+"<br>";
// 配列の15番目に代入(indexは14)
myArray[14] = true;
// 結果を出力
ele.innerHTML += "myArrayの要素数 :"+myArray.length+"<br>";
```

COLUMN

連想配列も含めた要素数を求めるには

配列オブジェクトの「length」プロパティは、添え字(数値)で指定される配列の要素数を返します。しかし、連想配列(ハッシュ)も含めた要素は対象外です。連想配列も含めた要素数を求めるには、次のように「for...in」文を利用します。

```
var ele = document.getElementsByTagName("output")[0];
// 配列を作成(6要素)
var myArray = ["A", "B", "C", 1, 2, 3];
// 配列をハッシュとして使用
myArray["MZ"] = 700;
myArray["PC"] = 6001;
myArray["FM"] = 7;
// ハッシュも含めて要素数を求める
var count = 0;
for(var i in myArray){ count++; }
// 配列の数を表示
ele.innerHTML = "myArrayの要素数 :"+myArray.length+"<br>";
ele.innerHTML += "myArrayの総要素数 :"+count+"<br>";
```

配列要素を加工する

ここでは、配列要素を加工する方法について解説します。

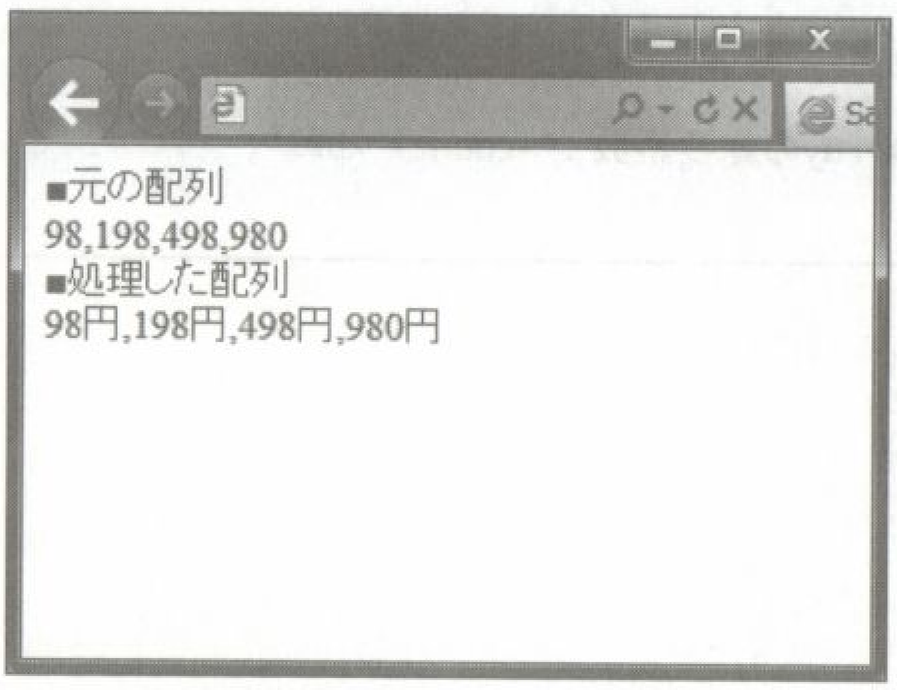
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = [98, 198, 498, 980];
  // 配列を順番に処理
  var ary = myArray.map(function(element, index, targetArray){
    // 読み出した要素に「円」の文字を末尾に追加
    return element+"円";
  });
  // 結果を出力
  ele.innerHTML = "■元の配列<br>"+myArray.toString()+"<br>";
  ele.innerHTML += "■処理した配列<br>"+ary.toString();
}, false);
```

ブラウザでindex.htmlを表示すると、要素の末尾に「円」の文字を追加した結果が表示されます。



ONEPOINT **配列要素を加工するには「map()」メソッドを使う**

配列要素を加工するには、「map()」メソッドを使います。「map()」メソッドのパラメータは、2つあります。最初のパラメータには、コールバック関数を指定します。2番目のパラメータには、コールバック関数内で「this」として使用するオブジェクトを指定します。なお、2番目のパラメータは省略することができます。

コールバック関数には、3つのパラメータが渡されます。最初のパラメータは配列の要素、2番目のパラメータには要素の位置、3番目は操作対象の配列になります。「map()」メソッドでは、1つずつ要素をコールバック関数に渡します。コールバック関数内では要素1つだけの処理を行い、結果を「return」文で返します。

「map()」メソッドは、すべての要素を処理した結果を新しい配列で返します。その際、元の配列内容は何も変化しません(非破壊)。

なお、古いブラウザでは「map()」メソッドは実装されていませんが、次のURLにあるコードを使うことで同等の機能を実装することができます。

URL https://developer.mozilla.org/ja/JavaScript/Reference/Global_Objects/Array/map

関連項目 ▶▶▶

● 配列要素を順番に処理する………………………………p.132

配列要素を追加・削除する

ここでは、配列要素を追加・削除する方法について解説します。

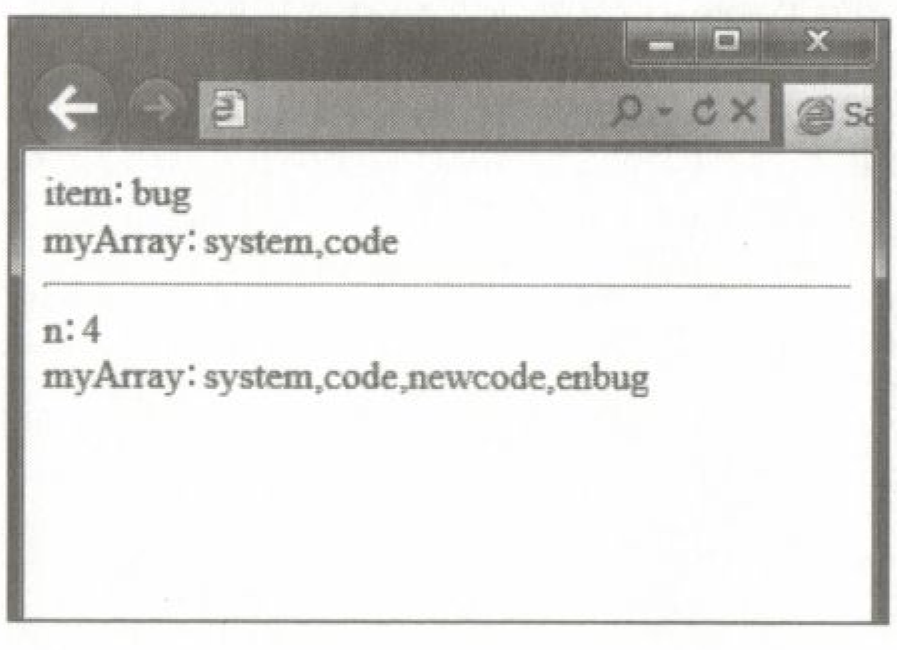
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["system", "code", "bug"];
  // 末尾の配列要素を取り出す
  var item = myArray.pop();
  // 結果を出力
  ele.innerHTML = "item:"+item+"<br>";
  ele.innerHTML += "myArray:"+myArray+"<hr>";
  // 配列の末尾に要素を追加する
  var n = myArray.push("newcode", "enbug");
  // 結果を出力
  ele.innerHTML += "n:"+n+"<br>";
  ele.innerHTML += "myArray:"+myArray;
}, false);
```

ブラウザでindex.htmlを表示すると、配列の末尾の要素を取り出した結果と、その後、配列末尾に要素を追加した結果が表示されます。



ONEPOINT 配列の末尾に要素を追加するには「push()」メソッド、末尾の要素の削除を行うには「pop()」メソッドを使う

配列に新たな要素を追加するには、「push()」メソッドを使います。「push()」メソッドのパラメータには、末尾に追加したい要素を指定します(複数指定可能)。「push()」メソッドは、追加された要素も含めた配列要素数を返します。

配列の末尾から要素を取り出すには、「pop()」メソッドを使います。取り出されるのは最後の要素1つだけです。「pop()」にはパラメータはありません。

COLUMN 任意の位置に要素を追加するには

JavaScriptでは、配列の任意の位置(インデックス)に要素を追加することができます。その際、配列のサイズを調整する必要はありません。自動的にブラウザ側でサイズが調整されます。また、次のように位置(インデックス)を指定した場合とハッシュ(連想配列)を指定した場合では、読み出し方法が異なる(配列サイズも異なる)ので注意が必要です。

```
var ele = document.getElementsByTagName("output")[0];
// 配列を作成
var myArray = [ ];
// 要素を任意の場所に追加
myArray[14] = "Up";
myArray[30] = "Down";
myArray["Quark"] = "RGB Color";
// 配列の内容を出力
ele.innerHTML = "■配列内容<br>"+myArray.toString()+"<br>";
// for...inで出力
for(var i in myArray){ ele.innerHTML += myArray[i] + "<br>"; }
```

関連項目 ▶▶▶

● 配列要素内容を前方/後方にずらす ………………………… p.149

配列要素を前方または後方から処理する

ここでは、配列要素を前方、または、後方から処理する方法について解説します。

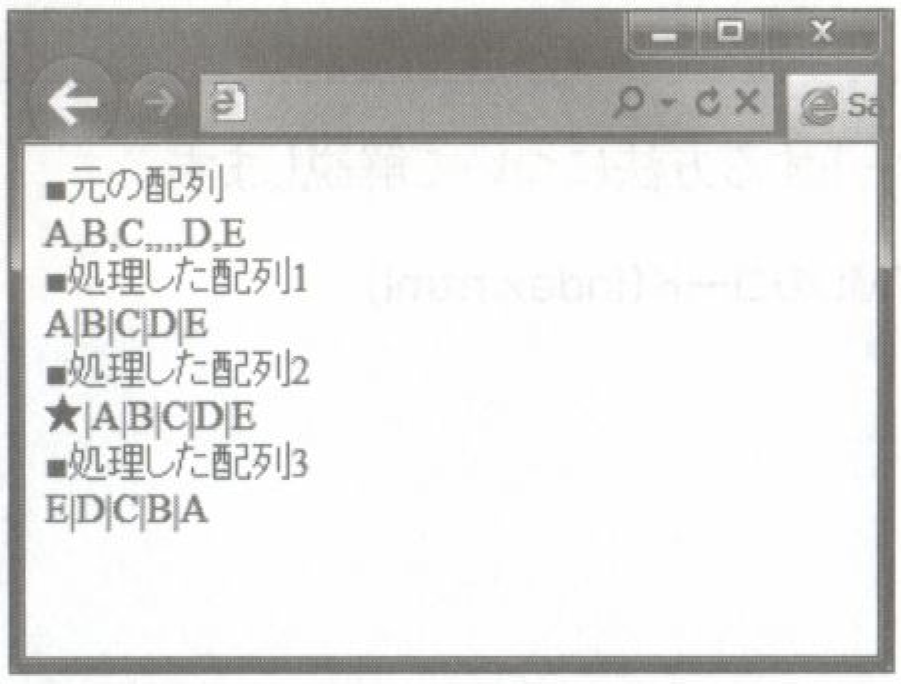
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["A", "B", "C", , , , "D", "E"];
  // 配列を前方から順番に処理
  var ary1 = myArray.reduce(connect);
  // 初期値を指定して配列を前方から順番に処理
  var ary2 = myArray.reduce(connect, "★");
  // 配列を後方から順番に処理
  var ary3 = myArray.reduceRight(connect);
  // 結果を出力
  ele.innerHTML = "■元の配列<br>"+myArray.toString()+"<br>";
  ele.innerHTML += "■処理した配列1<br>"+ary1.toString()+"<br>";
  ele.innerHTML += "■処理した配列2<br>"+ary2.toString()+"<br>";
  ele.innerHTML += "■処理した配列3<br>"+ary3.toString();
  // 内容を連結する関数
  function connect(prev, current, index, targetArray){
    // 前と現在の要素に|の文字を連結して返す
    return prev+"|"+current;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、要素ごとに先頭から「|」を連結した結果と末尾から連結した結果が表示されます。



ONEPOINT **要素を先頭から処理するには「reduce()」メソッド、末尾から処理するには「reduceRight()」メソッドを使う**

配列要素を先頭から処理するには、「reduce()」メソッドを使います。「reduce()」メソッドのパラメータは2つあります。最初のパラメータには、コールバック関数を指定します。2番目のパラメータには、初期値を指定します。なお、2番目のパラメータは省略することができます。

コールバック関数には、4つのパラメータが渡されます。最初のパラメータは前の値(コールバック関数で処理されてきた結果)、2番目のパラメータが現在の要素、3番目のパラメータには要素の位置、4番目は操作対象の配列になります。「reduce()」メソッドは、コールバック関数から「return」文で戻された内容を返します。

配列要素を末尾から処理する場合は、「reduceRight()」メソッドを使います。指定するパラメータやコールバック関数に渡される内容は、「reduce()」メソッドと同じです。

「reduce()」メソッド、「reduceRight()」メソッドとも配列の内容が疎(配列要素が空)の場合、その部分はコールバック関数には渡されません。また、元の配列は何も変化しません(非破壊)。

なお、古いブラウザでは「reduce()」メソッドと「reduceRight()」メソッドは実装されていませんが、次のURLにあるコードを使うことで同等の機能を実装することができます。

URL https://developer.mozilla.org/ja/JavaScript/Reference/Objects/Array/reduce

URL https://developer.mozilla.org/ja/JavaScript/Reference/Objects/Array/reduceRight

関連項目 ▶▶▶

● 配列要素を順番に処理する …… p.132
● 配列要素を加工する …… p.140

SECTION-050

配列要素をソートする

ここでは、配列要素をソートする方法について解説します。

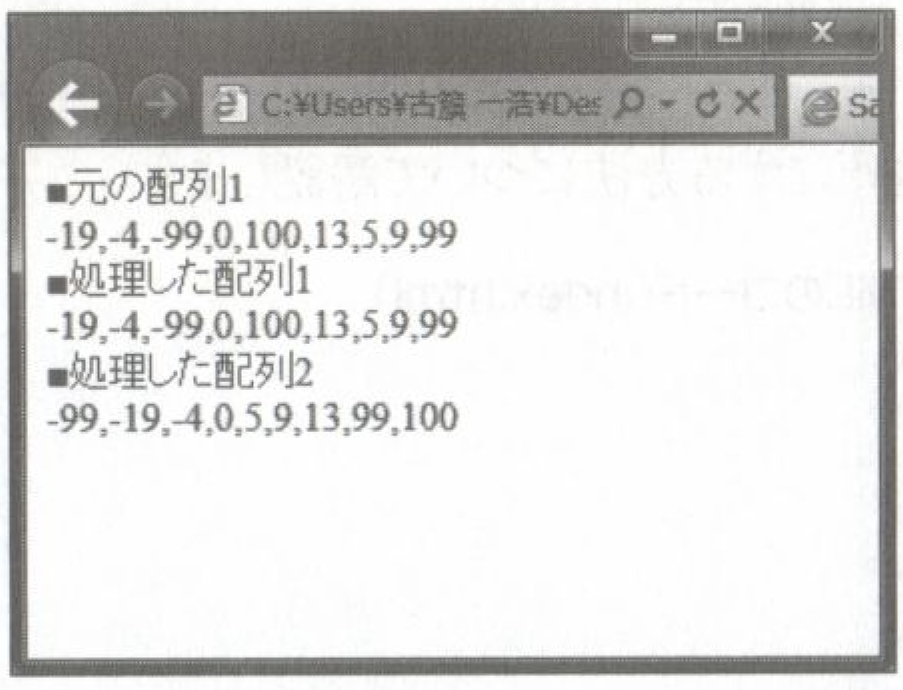
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = [0, -19, -4, 100, 5, 9, 13, 99, -99];
  var myArray2 = [0, -19, -4, 100, 5, 9, 13, 99, -99];
  // 配列をソート(文字コード単位)
  var ary1 = myArray1.sort();
  // 配列をソート関数を指定してソート
  var ary2 = myArray2.sort(function(a, b){
    return a - b;
  });
  // 結果を出力
  ele.innerHTML = "■元の配列1<br>"+myArray1.toString()+"<br>";
  ele.innerHTML += "■処理した配列1<br>"+ary1.toString()+"<br>";
  ele.innerHTML += "■処理した配列2<br>"+ary2.toString();
}, false);
```

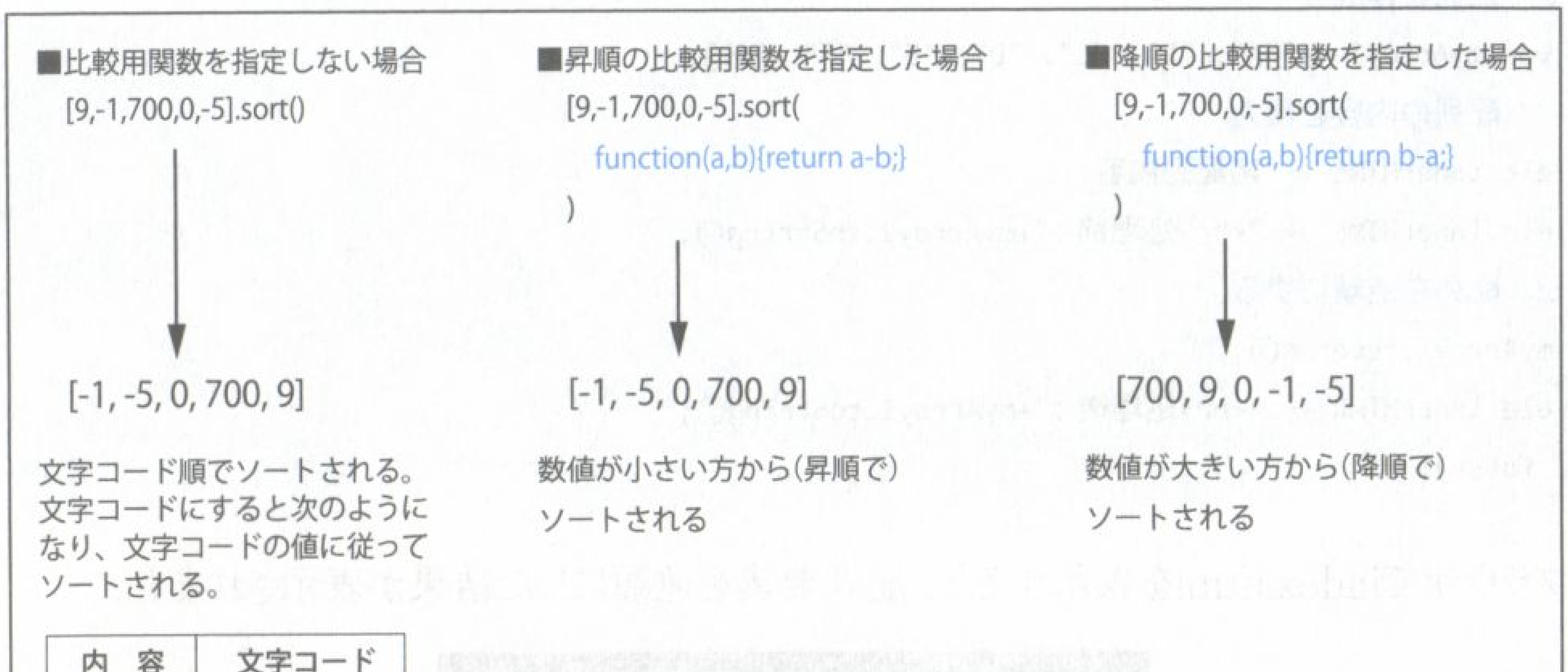
ブラウザでindex.htmlを表示すると、配列要素が文字コードでソートされた結果と、昇順でソートされた結果が表示されます。



ONEPOINT 要素をソートするには「sort()」メソッドを使う

要素をソートするには、「sort()」メソッドを使います。「sort()」メソッドのパラメータに何も指定しない場合、要素は文字コード順でソートされます。

パラメータには、比較用の関数を指定することができます。比較用の関数には2つのパラメータが渡され、戻り値によって要素がソートされます。戻り値が「0」未満の場合、昇順にソートされます。戻り値が「0」より大きい場合は、降順にソートされます。「0」の場合はソートされず、そのままになります。



内 容	文字コード
-1	45, 49
-5	45, 53
0	48
700	55, 48, 48
9	57

配列要素を逆順にする

ここでは、配列要素を逆順にする方法について解説します。

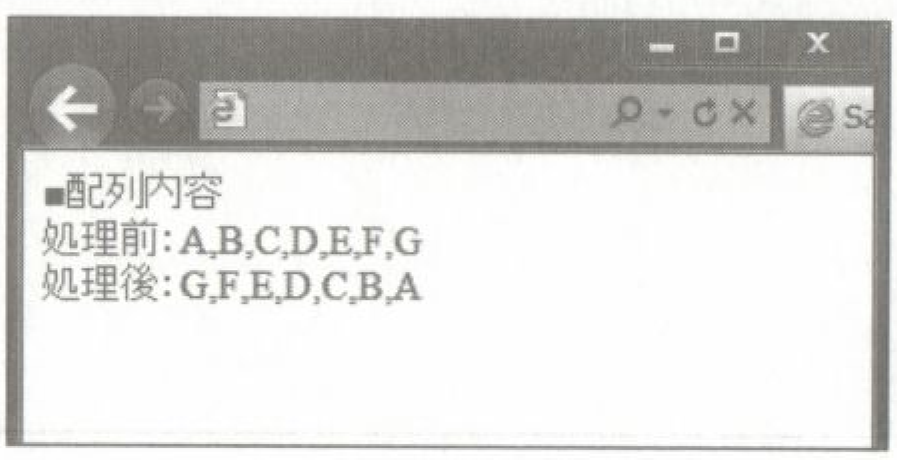
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray1 = ["A", "B", "C", "D", "E", "F", "G"];
  // 配列の内容を出力
  ele.innerHTML = "■配列内容";
  ele.innerHTML += "<br>処理前 :"+myArray1.toString();
  // 配列を逆順にする
  myArray1.reverse();
  ele.innerHTML += "<br>処理後 :"+myArray1.toString();
}, false);
```

ブラウザでindex.htmlを表示すると、配列要素を逆順にした結果が表示されます。



ONEPOINT 配列要素の順序を逆順にするには「reverse()」メソッドを使う

配列要素の順序を逆順にするには、「reverse()」メソッドを使います。「reverse()」メソッドを実行すると、指定した配列の要素そのものが逆順になります(配列は破壊)。また、「reverse()」メソッドは、逆順にした結果を配列として返します。

SECTION-052

配列要素内容を前方/後方にずらす

ここでは、配列要素内容を前方/後方にずらす方法について解説します。

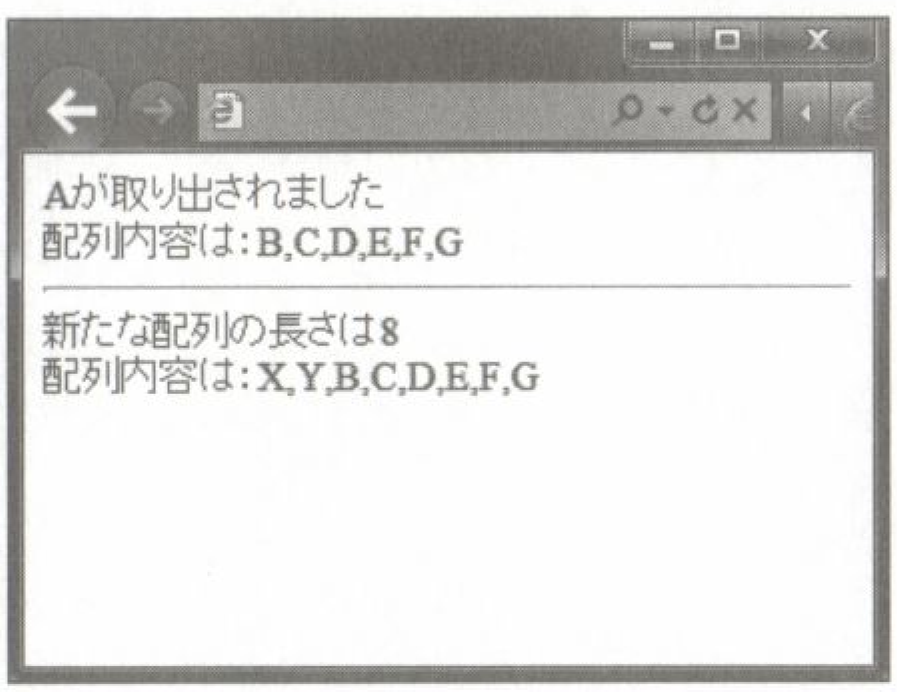
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["A", "B", "C", "D", "E", "F", "G"];
  // 配列の内容を前方にずらす
  var item = myArray.shift();
  ele.innerHTML = item + "が取り出されました<br>";
  ele.innerHTML += "配列内容は:"+myArray.toString();
  // 配列の内容を後方にずらし、新たな要素を追加
  var len = myArray.unshift("X", "Y");
  ele.innerHTML += "<hr>新たな配列の長さは"+len + "<br>";
  ele.innerHTML += "配列内容は:"+myArray.toString();
}, false);
```

ブラウザでindex.htmlを表示すると、配列の先頭の要素が取り除かれて要素が前方に移動した結果と、配列の先頭に要素を追加して要素が後方に移動した結果が表示されます。



ONEPOINT

配列内容を前方に移動させるには「shift()」メソッド、後方に移動させるには「unshift()」メソッドを使う

配列内容を前方に移動させるには、「shift()」メソッドを使います。「shift()」メソッドは、配列の最初の要素を取り出し、残りの要素を1つずつ前方にずらします。

配列内容を後方に移動させるには、「unshift()」メソッドを使います。「unshift()」メソッドには、配列の先頭に挿入したい内容を指定します。複数のパラメータを指定した場合は、その数だけ配列の先頭に挿入されます。また、「unshift()」メソッドは、配列に挿入後の全要素数を返します。

COLUMN

配列内容をローテーションさせる

配列内容をローテーションさせるには、次のコードのように「myArray.push(myArray.shift())」とします。取り出した要素を末尾に追加することで、配列内容をローテーションさせることができます。

```
var ele = document.getElementsByTagName("output")[0];
// 配列を作成
var myArray = ["A", "B", "C", "D", "E", "F", "G"];
// 配列の内容をローテーションさせる
for(var i=1; i<=10; i++){
  myArray.push(myArray.shift());
  ele.innerHTML += i+"回目の配列内容 :"+myArray.toString()+"<br>";
}
```

関連項目 ▶▶▶

●配列要素を追加・削除する……………………………………p.142

SECTION-053

配列要素を文字列に変換する

ここでは、配列要素を文字列に変換する方法について解説します。

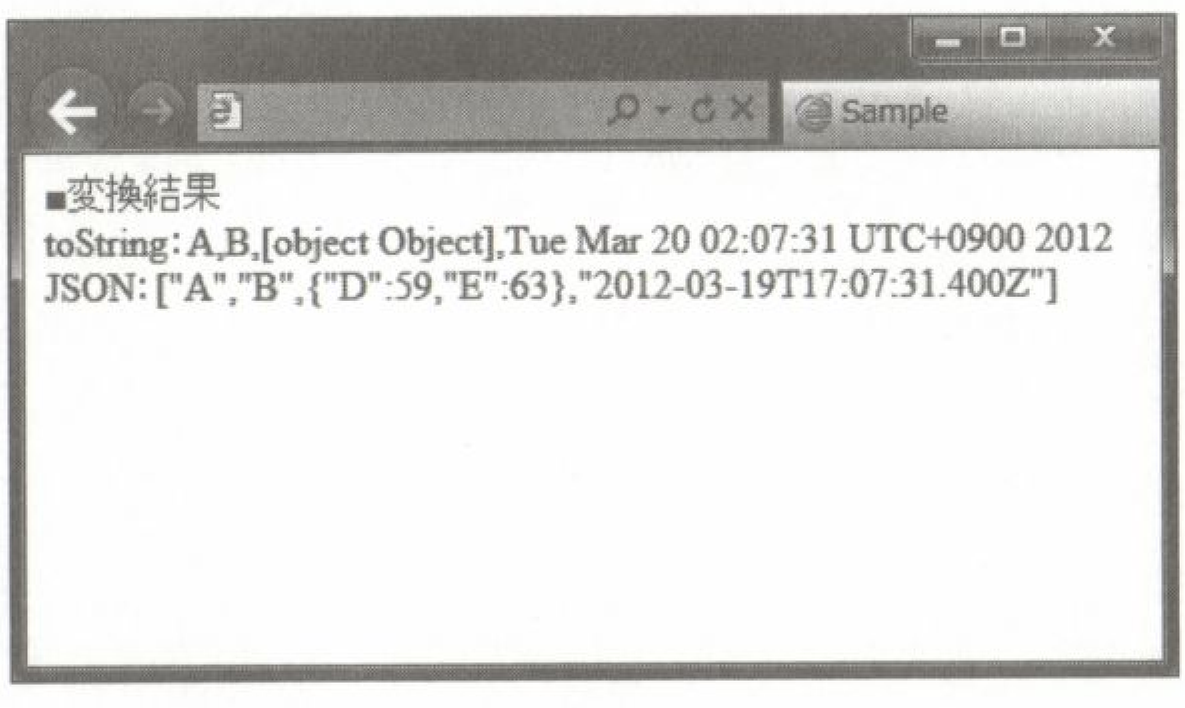
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 配列を作成
  var myArray = ["A", "B", { "D" : 59, "E" : 63}, new Date()];
  // 配列を文字列に変換する
  var myStr = myArray.toString();
  // 配列を文字列に変換する
  var myJson = JSON.stringify(myArray);
  // 配列の内容を出力
  ele.innerHTML = "■変換結果";
  ele.innerHTML += "<br>toString:"+myStr;
  ele.innerHTML += "<br>JSON:"+myJson;
}, false);
```

ブラウザでindex.htmlを表示すると、配列要素を文字列に変換した結果が表示されます。上の段は「toString()」メソッドを使って変換した結果で、下の段は「JSON.stringify()」メソッドを使って変換した結果となります。



ONEPOINT

配列要素を文字列に変換するには「toString()」メソッドや「JSON.stringify()」メソッドを使う

配列要素を文字列に変換するには、「toString()」メソッドを使います(配列オブジェクトの場合、「join()」メソッドを実行した結果と同じになる)。ただし、「toString()」メソッドは文字列や数値はそのまま処理しますが、オブジェクトの場合は「[object]」といった文字列に変換されてしまうことがあります。オブジェクトも変換したい場合には、「JSON.stringify()」メソッドを使う方法もあります。これは、オブジェクトをJSON形式に変換するメソッドになります。パラメータに変換したいオブジェクトを指定します。

COLUMN

配列オブジェクトの生成時間

配列オブジェクトを生成する場合、JavaScriptでは2種類の方法が用意されています。1つは「new Array()」とする方法で、もう1つが「[ ]」として配列を生成する方法です。高速に配列オブジェクトを生成したい場合には、「new Array()」でなく、「[ ]」を使います。ブラウザによっては数倍の速度差が発生する場合もありますが、「new Array()」でも「[ ]」でも大差がない場合もあります。いずれにせよ、「[ ]」を使った方が配列は高速に生成されます。

SECTION-054

「WeakMap」オブジェクトを生成する

ここでは、「WeakMap」オブジェクトを生成する方法について解説します。

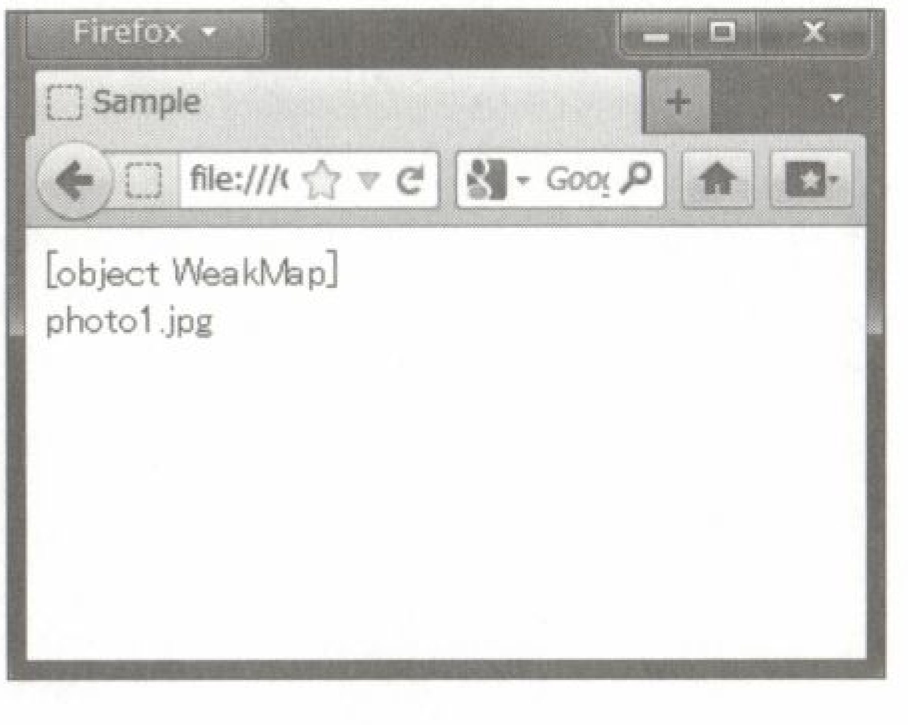
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WeakMapオブジェクトを生成
  var wmObj = new WeakMap();
  // キーにするオブジェクトを生成
  var obj = { model : "mz-721", price : 89800 };
  // Weakマップにキーと値を設定
  wmObj.set(obj, "photo1.jpg");
  // Weakマップの値を読み出し
  var data = wmObj.get(obj);
  // 結果を出力
  ele.innerHTML = wmObj.toString()+"<br>"+data;
}, false);
```

ブラウザでindex.htmlを表示すると、「WeakMap」オブジェクトであることを示す結果と、読み出した内容が表示されます。



ONEPOINT 「WeakMap」オブジェクトを生成するには「new WeakMap()」とする

JavaScriptでは、オブジェクト/連想配列(ハッシュ)を作成する際にキーと値を指定します。値にはほとんど制限がありませんが、キーにはオブジェクトが使用できないといった制限があります。キーにオブジェクトを使用したい場合には、「WeakMap」オブジェクトを利用します。

「WeakMap」オブジェクトを生成するには「new WeakMap()」とします。「WeakMap」オブジェクトでは、キーにオブジェクトが使用できるため、より柔軟な処理を行うことができます。生成した「WeakMap」オブジェクトに値を設定するには、「set()」メソッドを使います。最初のパラメータがキー、2番目のパラメータが値になります。設定した値を読み出す場合には、「get()」メソッドを使います。パラメータには、読み出したいキー名を指定します。「get()」メソッドは必要であれば、初期値を示す2番目のパラメータも指定することができます。

関連項目▶▶▶

- 配列オブジェクトを生成する …… p.120
- WeakMapで該当するキーがあるか調べる …… p.155
- WeakMapのキーを削除する …… p.157

SECTION-055

WeakMapで該当するキーがあるか調べる

ここでは、WeakMapで該当するキーがあるか調べる方法について解説します。

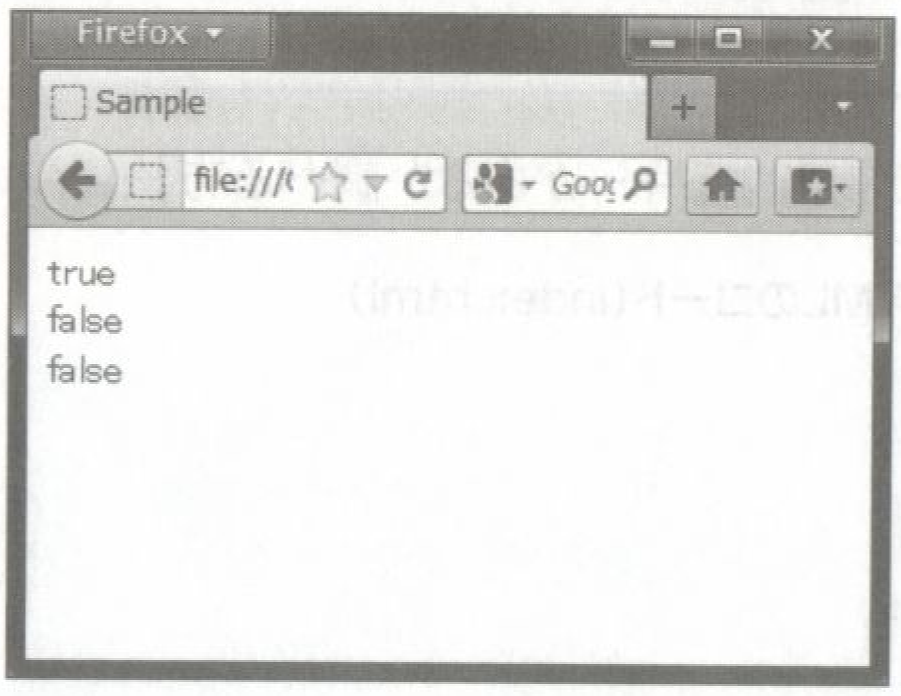
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WeakMapオブジェクトを生成
  var wmObj = new WeakMap();
  // キーにするオブジェクトを生成
  var obj1 = { model : "mz-711", price : 79800 };
  var obj2 = { model : "mz-721", price : 89800 };
  // Weakマップにキーと値を設定
  wmObj.set(obj1, "photo1.jpg");
  wmObj.set(obj2, "photo2.jpg");
  wmObj.set({model : "mz-731"}, "128000");
  // Weakマップに該当するキーがあるか調べる
  var flag1 = wmObj.has(obj1);
  var flag2 = wmObj.has(window);
  var flag3 = wmObj.has({model : "mz-731"});
  // 結果を出力
  ele.innerHTML = flag1+"<br>"+flag2+"<br>"+flag3;
}, false);
```

ブラウザでindex.htmlを表示すると、該当するキーがあるか調べ、その結果が表示されます。



ONEPOINT WeakMapでキーの有無を調べるには「has()」メソッドを使う

WeakMapで該当するキーが存在するかどうかは、「has()」メソッドで調べることができます。パラメータに、調べたいキーとなるオブジェクトを指定します。該当するキーがあれば「true」を返し、キーがなければ「false」を返します。

関連項目 ▶▶▶

● 「WeakMap」オブジェクトを生成する ………… p.153
● WeakMapのキーを削除する ………… p.157

SECTION-056

WeakMapのキーを削除する

ここでは、WeakMapのキーを削除する方法について解説します。

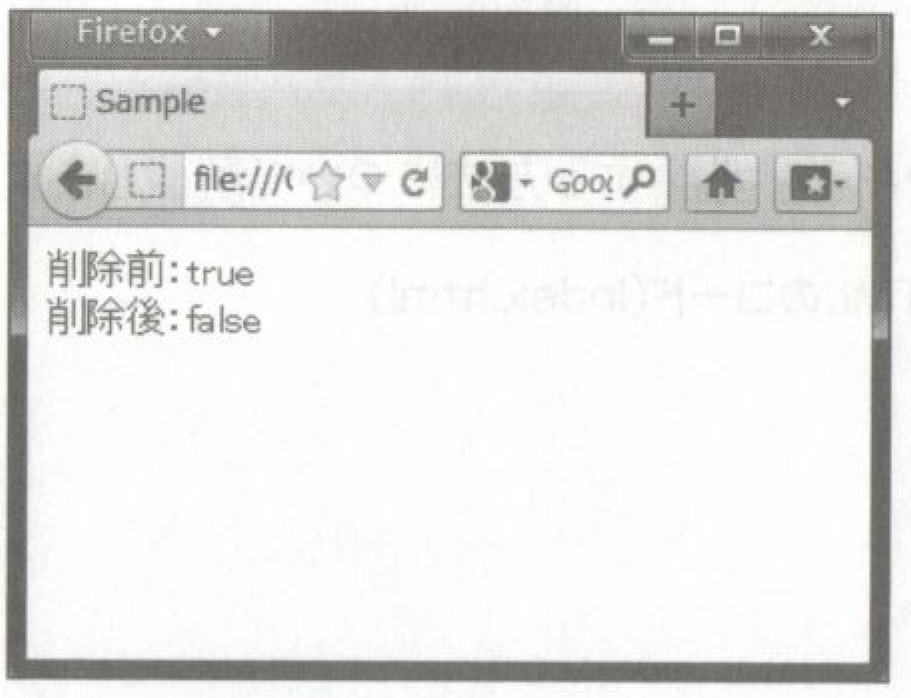
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WeakMapオブジェクトを生成
  var wmObj = new WeakMap();
  // キーにするオブジェクトを生成
  var obj = { model : "mz-711", price : 79800 };
  // Weakマップにキーと値を設定
  wmObj.set(obj, "photo1.jpg");
  // Weakマップに該当するキーがあるか調べる
  var flag = wmObj.has(obj);
  // 結果を出力
  ele.innerHTML = "削除前 :"+flag+"<br>";
  // キーを削除する
  wmObj.delete(obj);
  // Weakマップに該当するキーがあるか調べる
  flag = wmObj.has(obj);
  // 結果を出力
  ele.innerHTML += "削除後 :"+flag;
}, false);
```

ブラウザでindex.htmlを表示すると、削除する前はキーが存在していることを示す「true」となり、削除後はキーがないことを示す「false」が表示されています。



ONEPOINT キーの削除を行うには「delete()」メソッドを使う

WeakMapでキーを削除するには、「delete()」メソッドを使います。パラメータには、削除するキーとなるオブジェクトを指定します。パラメータに指定したオブジェクトが存在しない場合は、エラーになります。

関連項目 ▶▶▶

- 「WeakMap」オブジェクトを生成する ………… p.153
- WeakMapで該当するキーがあるか調べる ………… p.155

CHAPTER 04

数値・数学

「Number」オブジェクトを生成する

ここでは、「Number」オブジェクトを生成する方法について解説します。

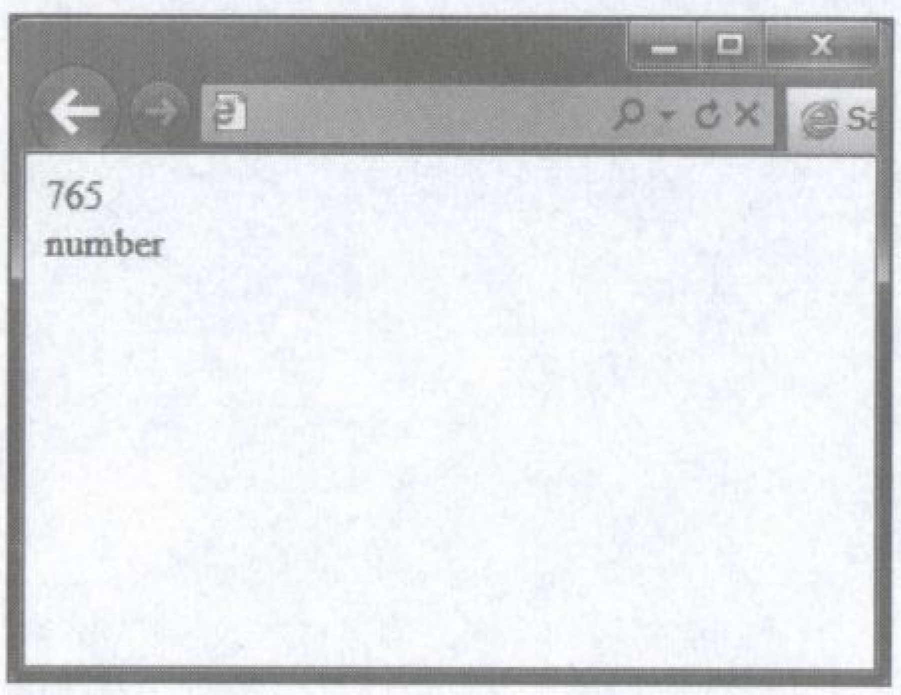
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数nを数値オブジェクトとして生成
  var n = new Number("764");
  // 値に1加算する
  n = n + 1;
  // 結果を出力
  ele.innerHTML = n+"<br>"+typeof(n);
}, false);
```

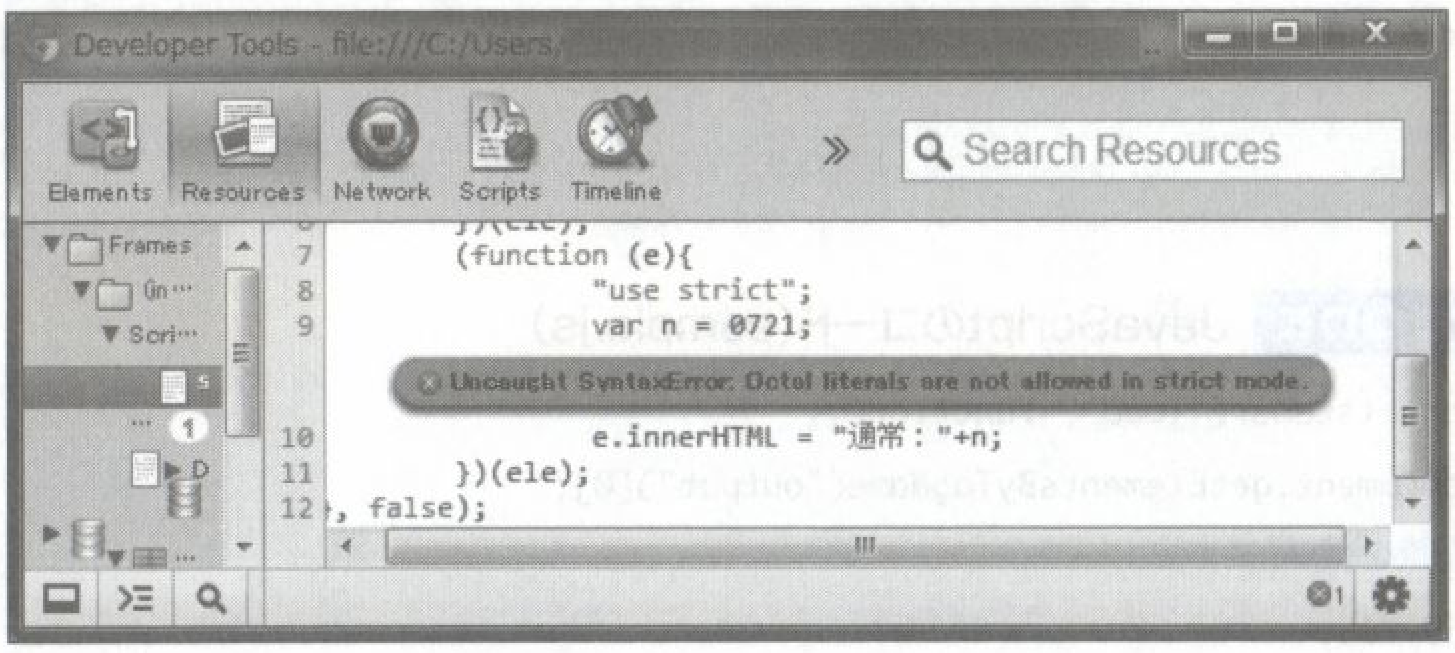
ブラウザでindex.htmlを表示すると、「Number」オブジェクトで生成された値に加算した結果と変数の種類が表示されます。



ONEPOINT 「Number」オブジェクトを生成するには「new Number()」とする

JavaScriptで数値型のオブジェクトを生成するには、「new Number()」とします。パラメータに何も指定しない場合には、変数は数値型として生成されます。パラメータを指定した場合、数値に変換可能なものであれば数値に変換されます。そうでない場合は、数値ではないことを示す「NaN」(Not a Number)になります。

JavaScriptでは8進数、10進数、16進数を扱うことができます。8進数は先頭が「0」から始まり、数値だけで構成される値になります。たとえば、「053」や「011」です。10進数は「0」から始まらない数値になります。16進数は先頭2文字が「0x」で始まり、「0」～「9」と「a」～「f」(大文字小文字は関係なし)で構成される値になります。たとえば、「0xff」や「0x96ba」です。なお、「strict」モードの場合、8進数を扱うことはできず、「0」から始まる数値を指定するとエラーが発生します(下図参照)。



COLUMN 虚数について

JavaScriptでは実数のみを扱うことができ、標準のオブジェクトで虚数を扱う方法は用意されていません。

関連項目 ▶▶▶

● 大きな数値同士の演算を行う……………………………………p.173

SECTION-058

数値の桁数を指定する

ここでは、数値の桁数を指定する方法について解説します。

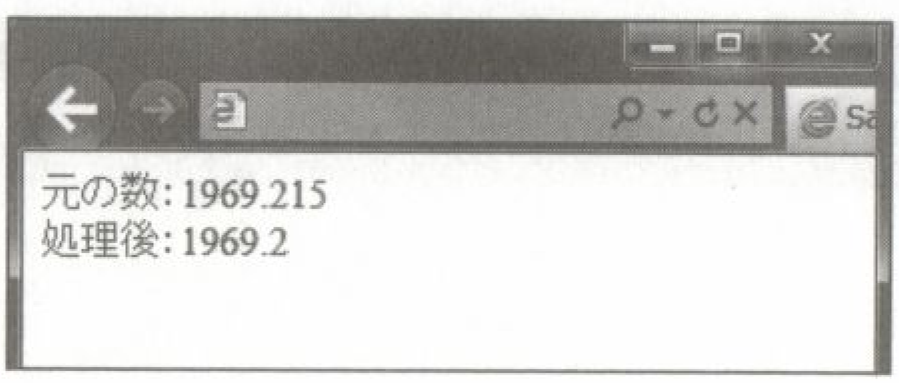
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数に数値を代入
  var n = 1969.215;
  // 桁数を指定する
  var num = n.toPrecision(5);
  // 結果を出力
  ele.innerHTML = "元の数：“+n+"<br>処理後：“+num;
}, false);
```

ブラウザでindex.htmlを表示すると、数値に桁数を指定した結果が表示されます。



ONEPOINT 桁数を指定するには「toPrecision()」メソッドを使う

JavaScriptで数値の桁数を指定するには、「toPrecision()」メソッドを使います。たとえば、「12345.678」という数値があり、「toPrecision(6)」とすると「12345.7」が返されます。「toPrecision(2)」とすると、「1.2e+4」のように指数形式で返されます。

関連項目 ▶▶▶

● 数値を小数点以下n桁にする ……………………………… p.163

数値を小数点以下n桁にする

ここでは、数値を小数点以下n桁にする方法について解説します。

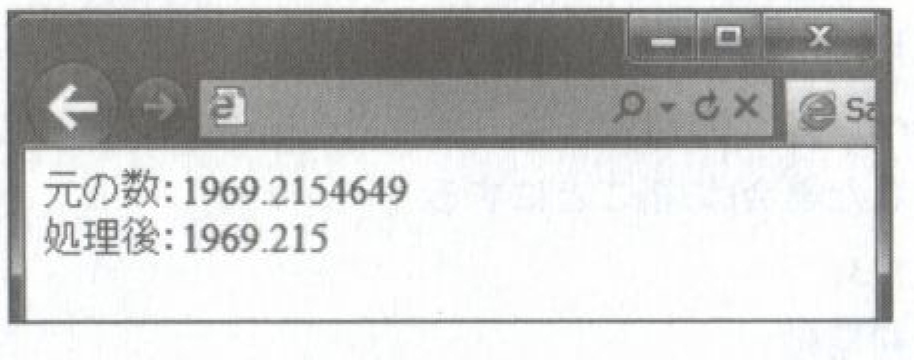
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var n = 1969.2154649;  // 変数に数値を代入
  var num = n.toFixed(3);  // 小数点以下を3桁にする
  ele.innerHTML = "元の数 :"+n+"<br>処理後 :"+num;  // 結果を出力
}, false);
```

ブラウザでindex.htmlを表示すると、指定した小数点以下の桁数にした数値が表示されます。



ONEPOINT 小数点以下の桁数を指定するには「toFixed()」メソッドを使う

JavaScriptで数値の小数点以下の桁数を指定するには、「toFixed()」メソッドを使います。たとえば、「12345.6789」という数値があり、「toFixed(2)」とすると、12345.68が返されます(指定した桁数以下は四捨五入される)。小数値を四捨五入して整数化したい場合には、「toFixed(0)」とします。

関連項目 ▶▶▶

●数値の桁数を指定する …… p.162
●四捨五入/切り捨て/切り上げを行う …… p.184

3桁ごとに区切る

ここでは、数値を3桁ごとに区切る方法について解説します。

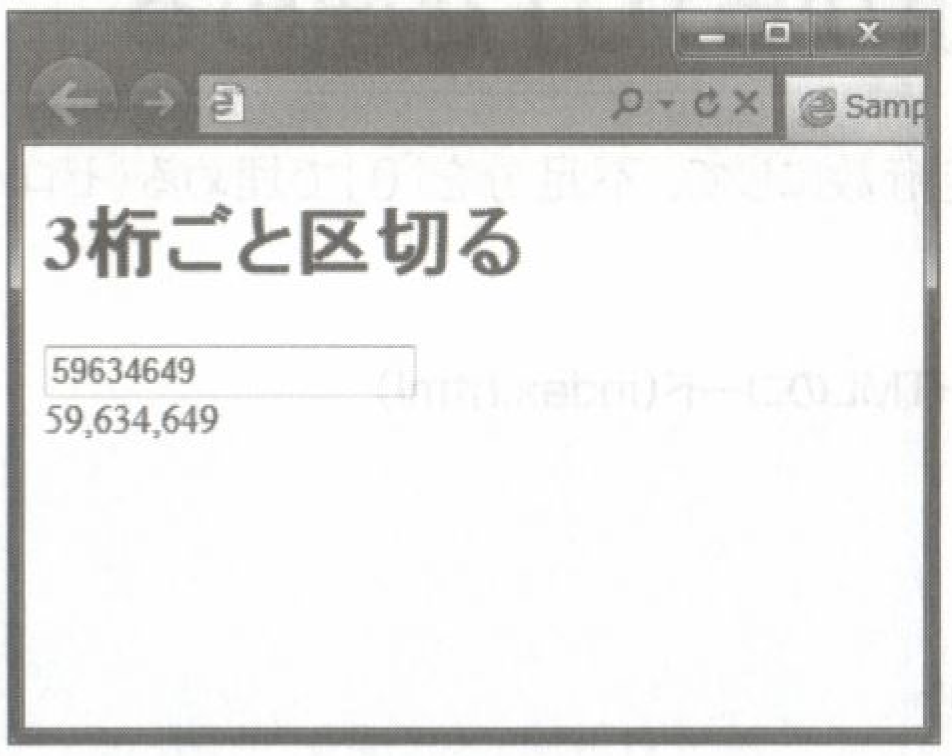
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>3桁ごと区切る</h1>
    <form>
      <input type="text" id="num" value="5963">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // テキストフィールドに入力されたらリアルタイムに結果を表示
  document.getElementById("num").onkeyup = function(){
    ele.innerHTML = num3(this.value);
  }
  // 指定した桁数ごとに区切った結果を返す関数
  function num3(num, separate){
    // パラメータが省略された場合は3桁ごとにする
    separate = separate || 3;
    // 結果を入れる変数を初期化
    var n = "";
    var count = 0;
    // 数値を文字列に変換
    var str = new String(num);
    for (var i=str.length-1; i>=0; i--){
      n = str.charAt(i) + n;
      count++;
      if (((count % separate) == 0) && (i != 0)) n = ","+n;
    }
    return n;
  }
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに数値を入力すると、リアルタイムに3桁に区切られた数値が表示されます。



ONEPOINT　3桁ごとに区切るには数値を文字列に変換して「,」を連結する

3桁ごとに区切るには、数値を一度、文字型に変換します。変換した文字列を末尾から連結、および、カウントしていき、「3」で割ります。その余りが「0」の場合に、「,」(カンマ)を文字列に連結します。これを数値の桁数(文字列の長さ)だけ繰り返します。サンプルでは、「num3」関数内の2番目のパラメータに「,」(カンマ)で区切る桁数を指定できます。「4」にすると4桁ごとに「,」(カンマ)で区切られます。

SECTION-061

数値を指定桁数にして不足分を先頭から「0」で埋める

ここでは、数値を指定桁数にして、不足分を「0」で埋める(ゼロパディング)方法について解説します。

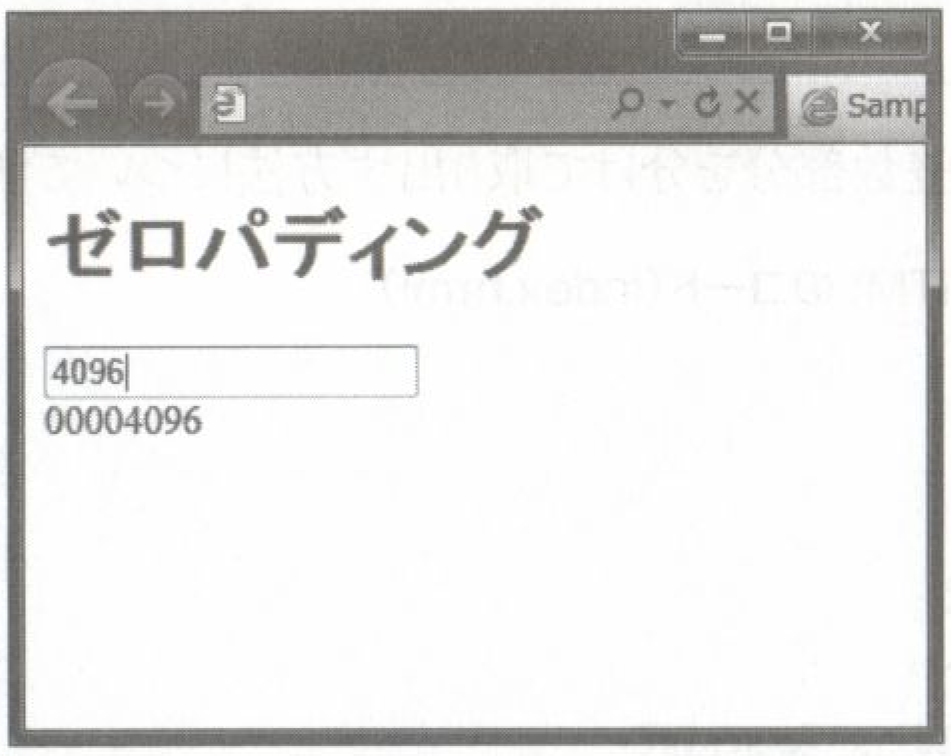
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ゼロパディング</h1>
    <form>
      <input type="text" id="num" value="256">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // テキストフィールドに入力されたらリアルタイムに結果を表示
  document.getElementById("num").onkeyup = function(){
    ele.innerHTML = zeroPadding(this.value, 8);
  }
  // 指定した桁数に揃える。不足分は0で埋める
  function zeroPadding(num, count){
    // 2番目のパラメータが省略された場合は8桁にする
    count = count || 8;
    // ゼロの文字を指定した桁数だけ生成
    var zero = "";
    for(var i=0; i<count; i++){
      zero = zero + "0";
    }
    // 0で埋める
    var n = (zero+num).slice(-count);
    return n;
  }
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに数値を入力すると、リアルタイムに数値が8桁で表示されます。8桁に満たない場合は、先頭が「0」で埋められます。



ONEPOINT **不足分の桁を「0」で埋めるには「slice()」メソッドを使う**

数値を指定した桁数で揃える場合には、「slice()」メソッドを使います。「slice()」メソッドは、指定した数だけ文字列を抜き出すことができます。その際に負数を指定すると、文字列の右側から指定した文字数を抜き出します。「0」で構成された文字列と連結した後で「slice()」メソッドを使うことで、指定した桁数に揃えることができます。

SECTION-062

小数部分と整数部分を分けて取り出す

ここでは、小数部分と整数部分を分けて取り出す方法について解説します。

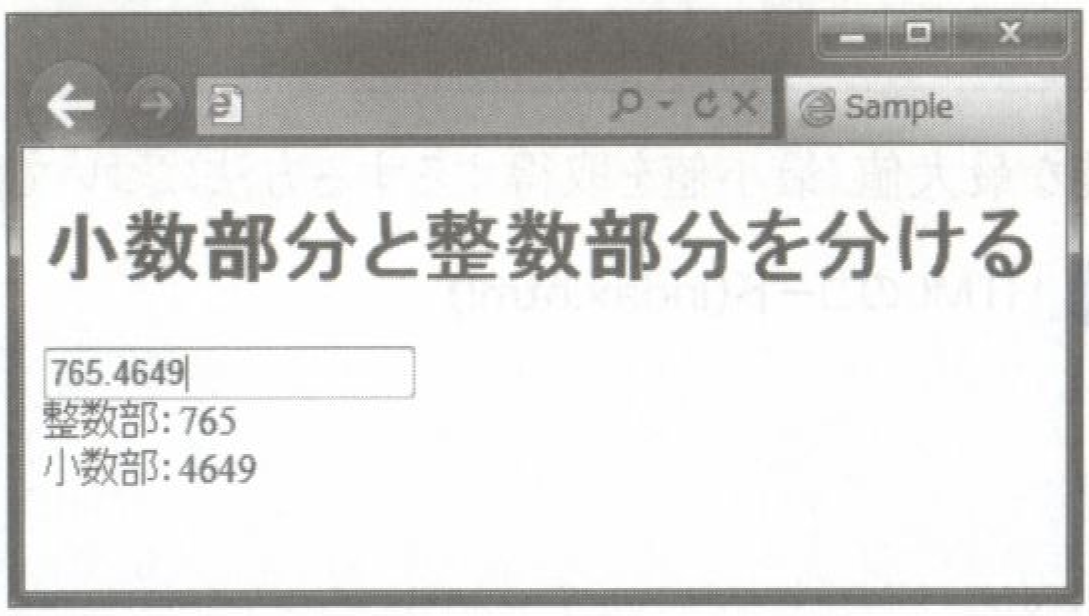
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>小数部分と整数部分を分ける</h1>
    <form>
      <input type="text" id="num" value="987.456">
    </form>
    <div>整数部：<output></output></div>
    <div>小数部：<output></output></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // テキストフィールドに入力されたらリアルタイムに結果を表示
  document.getElementById("num").onkeyup = function(){
    var n = separate(this.value);
    ele[0].innerHTML = n.integer;
    ele[1].innerHTML = n.float;
  }
  // 整数部と小数部を切り分ける
  function separate(num){
    // ゼロの文字を指定した桁数だけ生成
    var data = (""+num).split(".");
    var integerN = parseInt(data[0]);
    var floatN = parseInt(data[1]);
    return { integer : integerN, float : floatN }
  }
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに数値を入力すると、リアルタイムに整数部と小数部が表示されます。



ONEPOINT **小数部分と整数部分を分けるには文字型に変換した後で「split()」メソッドで分割する**

数値を小数部分と整数部分を分けるには、一度、数値を文字型に変換します。文字型に変換した後で「split()」メソッドを使ってピリオドを区切りとして分割します。分割すると配列が作成され、最初の要素に整数部が、2番目の要素に小数部が入ります。得られた結果を「praseInt()」メソッドなどを使って数値に変換します。

SECTION-063

使用可能な最大値/最小値を取得する

ここでは、使用可能な最大値/最小値を取得するする方法について解説します。

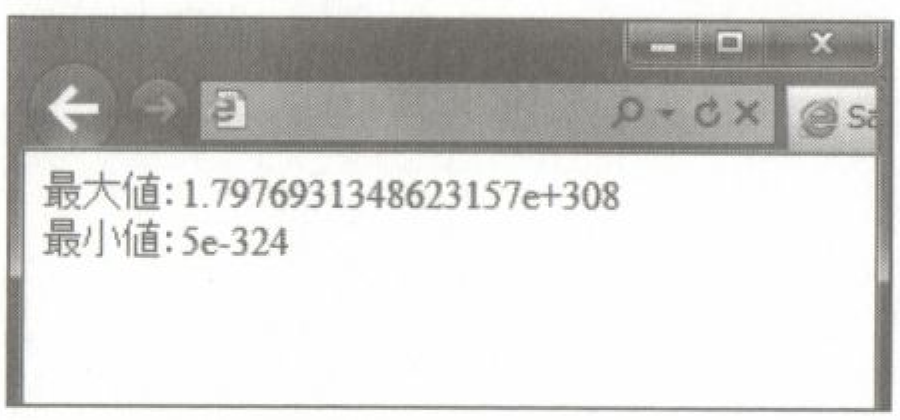
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 変数maxに最大値を代入
  var max = Number.MAX_VALUE;
  // 変数minに最小値を代入
  var min = Number.MIN_VALUE;
  // 結果を出力
  ele.innerHTML = "最大値 :"+max+"<br>最小値 :"+min;
}, false);
```

ブラウザでindex.htmlを表示すると、扱える最大値と最小値が表示されます。



ONEPOINT **扱える値の最大値を取得するには「Number.MAX_VALUE」、最小値を取得するには「Number.MIN_VALUE」を使う**

JavaScriptでは、実数値は64ビットで処理されます(仮数部52ビット、指数部11ビット、符号1ビット)。扱える値の最大値は「Number.MAX_VALUE」、最小値は「Number.MIN_VALUE」として定義されています。

また、正の無限大は「Number.POSITIVE_INFINITY」、負の無限大は「Number.NEGATIVE_INFINITY」として定義されています。

SECTION-064

演算誤差を減らす

JavaScriptで小数演算を行うと、誤差が発生して期待する精度が得られない場合があります。ここでは、演算誤差を減らす方法について解説します。

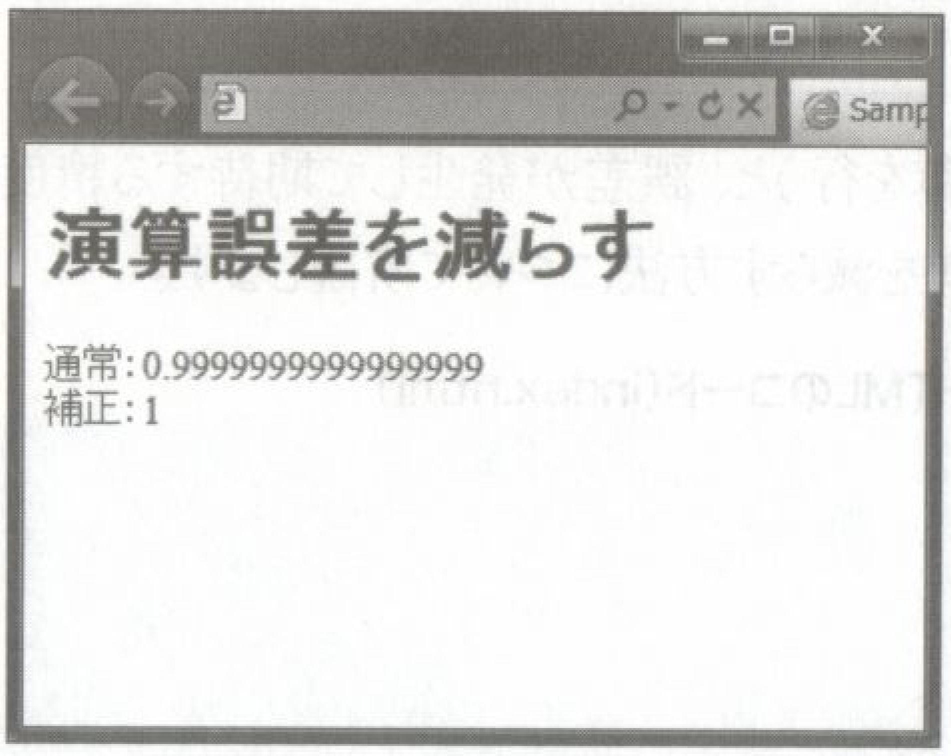
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>演算誤差を減らす</h1>
    <div>通常：<output></output></div>
    <div>補正：<output></output></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // ゲタをはかせずに0.1を10回加算する
  var num1 = 0;
  for(var i=0; i<10; i++){
    // 0.1の場合、加算すると誤差発生
    num1 = num1 + 0.1;
  }
  ele[0].innerHTML = num1;
  // ゲタをはかせて0.1を10回加算する
  var jackup = 10;  // 10倍。10のゲタをはかせる
  var num2 = 0;
  for(i=0; i<10; i++){
    // 0.1の場合、加算すると誤差発生
    num2 = num2 + 0.1*jackup;
  }
  // 10倍したので最後に10で除算する
  num2 = num2 / jackup;
  ele[1].innerHTML = num2;
}, false);
```

ブラウザでindex.htmlを表示すると、「0.1」を10回、加算した結果が表示されます。通常の演算では誤差が発生して「1」になりませんが、補正した場合は「1」になります。



ONEPOINT　演算誤差を減らすにはいったん整数にして演算する

JavaScriptに限らず多くのコンピュータ言語やCPUでは、浮動小数演算を行うと誤差が発生します。これは小数が2の-n乗の和で表現されるためで、値によっては表現することができないためです。これを軽減させるには、小数値を10倍するなどしてゲタをはかせます。整数演算の場合は小数演算と異なり、2のn乗で表現できるため、誤差が発生しません。

なお、演算誤差については、次のURLを参照してください。

URL http://itpro.nikkeibp.co.jp/article/COLUMN/20071019/285010/

また、数値計算を行うBigDecimal.jsライブラリを利用することでも演算誤差を防ぐことができます（BigDecimal.jsライブラリは何種類かある）。BigDecimal.jsライブラリを利用したサンプルについては、173ページを参照してください。

URL http://stz-ida.de/index.php?option=com_remository&Itemid=48&func=startdown&id=13

URL https://github.com/dtrebbien/BigDecimal.js

URL https://github.com/iriscouch/bigdecimal.js

関連項目 ▶▶▶

● 大きな数値同士の演算を行う……p.173

SECTION-065

大きな数値同士の演算を行う

JavaScriptの演算範囲を超えた範囲の数値演算を行うには、BigDecimal.jsライブラリを利用します。ここでは、BigDecimal.jsライブラリを使って数値演算を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

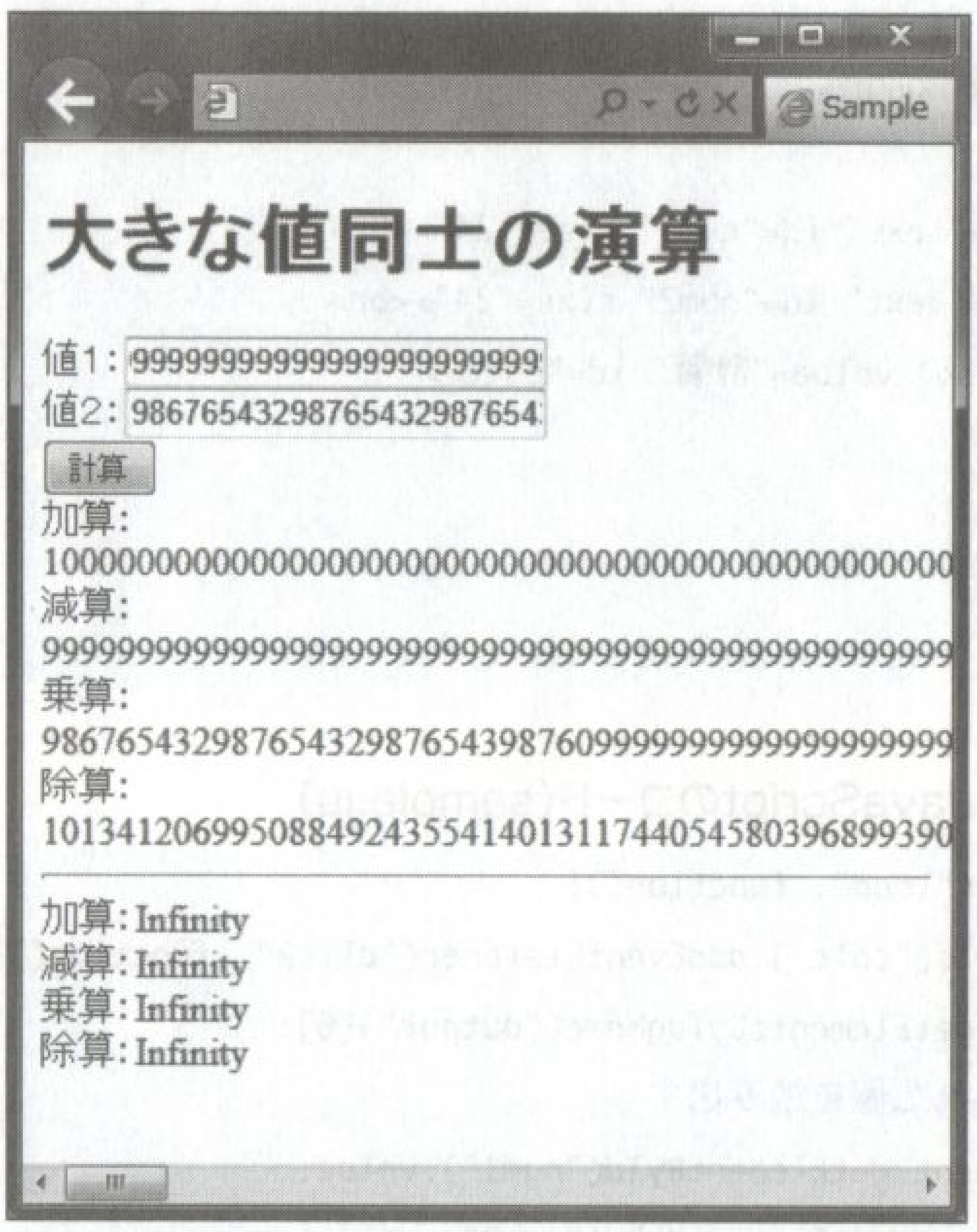
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/BigDecimal-all-last.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>大きな値同士の演算</h1>
    <form>
      値1 :<input type="text" id="num1" size="24"><br>
      値2 :<input type="text" id="num2" size="24"><br>
      <input type="button" value="計算" id="calc">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

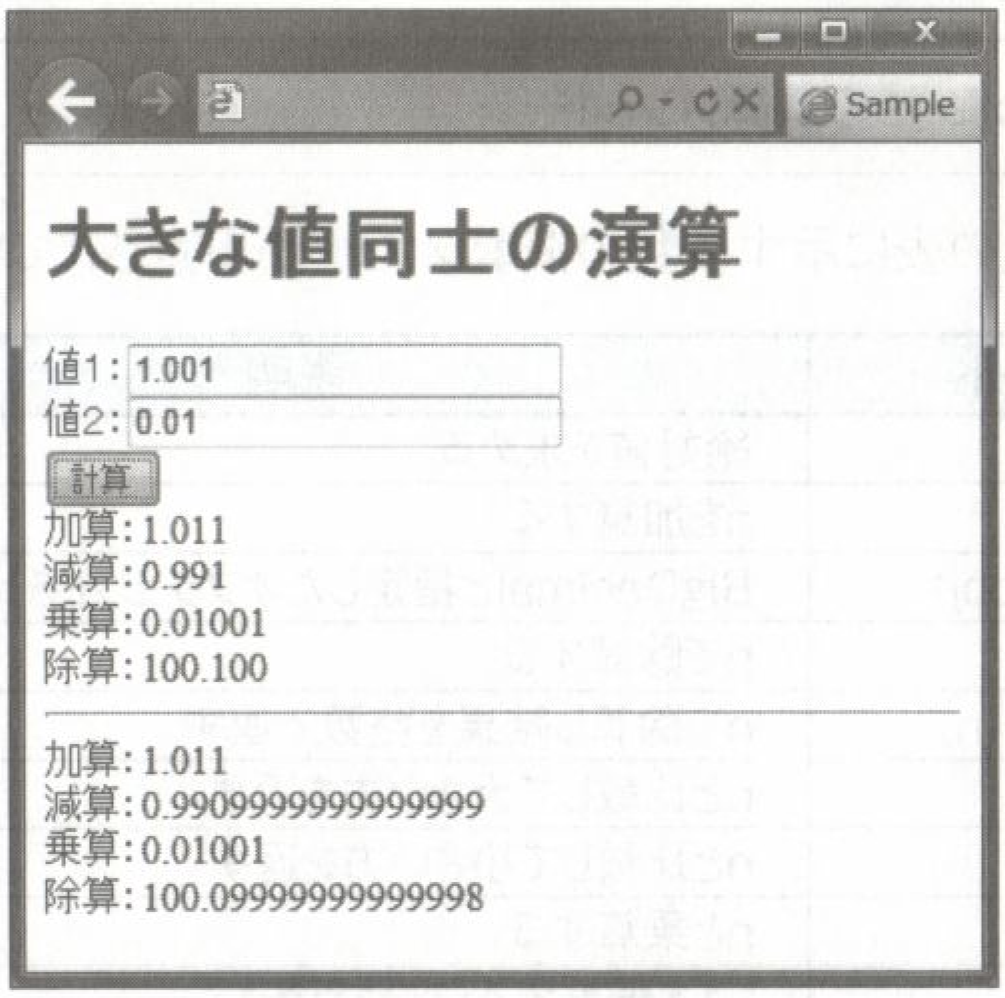
```
window.addEventListener("load", function(){
  document.getElementById("calc").addEventListener("click", function(){
    var ele = document.getElementsByTagName("output")[0];
    // フォームに入力された値を読み出す
    var number1 = document.getElementById("num1").value;
    var number2 = document.getElementById("num2").value;
    // BigDecimalオブジェクトにする(パラメータは文字列)
    var n1 = new BigDecimal(number1);
    var n2 = new BigDecimal(number2);
    // 加算した結果を表示
    var n = n1.add(n2);
    ele.innerHTML = "加算 :"+n+"<br>";
    // 減算した結果を表示
    n = n1.subtract(n2);
    ele.innerHTML += "減算 :"+n+"<br>";
    // 乗算した結果を表示
    n = n1.multiply(n2);
```

```
        ele.innerHTML += "乗算：" + n + "<br>";
        // 除算した結果を表示
        n = n1.divide(n2);
        ele.innerHTML += "除算：" + n + "<hr>";
        // 通常のJavaScriptでの演算結果
        number1 = parseFloat(number1);
        number2 = parseFloat(number2);
        ele.innerHTML += "加算：" + (number1+number2) + "<br>";
        ele.innerHTML += "減算：" + (number1-number2) + "<br>";
        ele.innerHTML += "乗算：" + (number1*number2) + "<br>";
        ele.innerHTML += "除算：" + (number1/number2) + "<br>";
    }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、大きな数値同士で四則演算を行った結果が表示されます。BigDecimal.jsライブラリを使った場合は計算結果が表示されますが、通常のJavaScriptでの計算は無限大となります。



BigDecimal.jsライブラリを使用した場合、演算誤差も軽減されます。通常のJavaScriptでは正しい結果が得られませんが、BigDecimal.jsを使うと正しい結果を得ることができます。



ONEPOINT 正しい演算結果を得るにはBigDecimal.jsライブラリを利用する

JavaScriptでの数値演算は、IEEE 754（浮動小数点数演算標準）に従って行われます。これは、一般的なコンピュータ上で動作する言語と同じです。IEEE 754は倍精度（64ビット）での浮動小数演算を行うため、その範囲を超える数値を取り扱うことができません。JavaScriptでは範囲を超えた場合、無限大を示す「Infinity」、または、負の無限大を示す「-Infinity」となってしまいます。また、小数演算を行うと演算誤差も発生するため、正しい計算結果を求められる処理では問題になることがあります。

このような場合、BigDecimal.jsライブラリを利用します。元は他の言語のもので、BigDecimal.jsはJavaScriptに移植されたものになります。BigDecimal.jsライブラリはいくつかありますが、ここでは、次のURLに示すBigDecimal.jsライブラリを使用しています。

- Big Decimal

  URL https://github.com/dtrebbien/BigDecimal.js

このBigDecimal.jsライブラリでは、「new BigDecimal()」として大きな数値を扱うオブジェクトを生成します。その際、パラメータには数値ではなく、文字列を指定します。つまり、「123.4」であれば、「"123.4"」のように指定します。また、計算結果で小数値を得たい場合には「"5963.0"」のように「.0」を付加して小数値であることを明示する必要があります。

BigDecimal.jsライブラリで四則演算を行うメソッドは、次の表のようになります。

メソッド	説明
add()	加算
subtract()	減算
multiply()	乗算
divide()	除算

なお、IEEE 754については、次のURLを参照してください。

URL http://ja.wikipedia.org/wiki/IEEE_754

COLUMN BigDecila.jsの演算メソッド

BigDecila.jsには、次の表に示す多数の演算メソッドが用意されています。

演算メソッド	説明
abs()	絶対値を求める
add(n)	nを加算する
compareTo(obj)	BigDecimalと指定したオブジェクトを比較する
divide(n)	nで除算する
divideInteger(n)	nで除算し結果を整数で返す
max(n)	nと比較して大きい方を返す
min(n)	nと比較して小さい方を返す
multiply(n)	nと乗算する
negate()	−1を乗算する(符号反転)
plus()	+1を乗算する(値に変化なし)
pow(n)	値のn乗を返す
remainder(n)	nと除算した余りを返す(商を返す)
subtract(n)	nを減算する
equals(n)	nと等しいか調べる
intValueExact()	整数に変換する
movePointLeft(n)	小数点を左へn桁移動する
movePointRight(n)	小数点を右へn桁移動する
scale()	スケールを返す
setScale(n)	スケールを設定する
signum()	符号を返す
toString()	文字列に変換する
round()	四捨五入する

関連項目 ▶▶▶

● 演算誤差を減らす ……………………………………………… p.171

SECTION-066

ビット演算を行う

ここでは、JavaScriptでビット演算を行う方法について解説します。

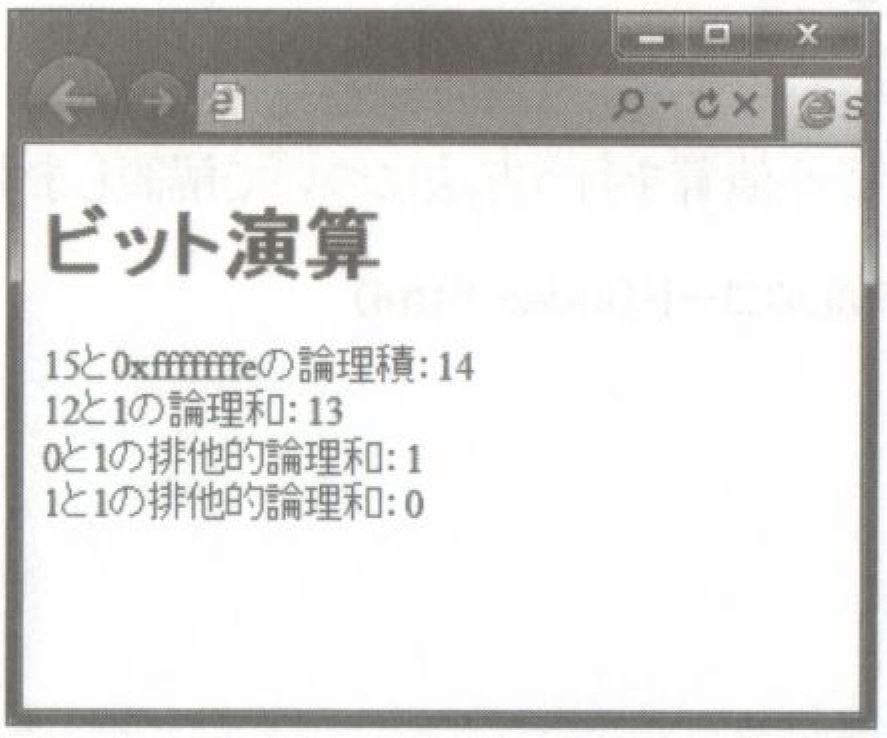
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ビット演算</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 処理する数値を指定
  var n = 15;
  // 論理積を使って数値を偶数にする
  var num1 = n & 0xfffffffe;
  // 結果を表示
  ele.innerHTML = n+"と0xfffffffeの論理積 :"+num1+"<br>";
  // 処理する数値を指定
  n = 12;
  // 論理和を使って数値を奇数にする
  var num2 = n | 1;
  // 結果を表示
  ele.innerHTML += n+"と1の論理和 :"+num2+"<br>";
  // 処理する数値を指定
  n = 0;
  // 排他的論理和を使って数値のビットを反転させ結果を表示
  var num3 = n ^ 1;
  ele.innerHTML += n+"と1の排他的論理和 :"+num3+"<br>";
  // 排他的論理和を使って数値のビットを反転させ結果を表示
  var num4 = num3 ^ 1;
  ele.innerHTML += num3+"と1の排他的論理和 :"+num4+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、数値をビット演算（論理積、論理和、排他的論理和）した結果が表示されます。



ONEPOINT　ビット演算を行うにはビット演算子を使う

JavaScriptでは、32ビット長でのビット演算を行うことができます。ビット演算は数値を「0」と「1」の2進数にし、各ビットに対して処理を行います。それぞれのビット演算の演算子と処理内容を次の表に示します。

演算子	処理内容	説明
&	論理積	どちらかのビットが「0」なら「0」
|	論理和	どちらかのビットが「1」なら「1」
^	排他的論理和	同じビット値なら「0」、そうでなければ「1」

論理積の簡単な使用例としては、値を常に偶数にすることができます。サンプルのように「0xfffffffe」と論理積を取れば最下位ビットが必ず「0」になるため、値は常に偶数になります。また、「0x0000000f」（15）と論理積を取れば、値は常に「0」～「15」の範囲に収まるようになります。

論理和は数値を常に奇数にすることができます。値と「1」との論理和を取れば常に最下位ビットが「1」になるため、常に奇数になります。

排他的論理和を利用すると、値を交互に切り換えることができます。たとえば、値が「0」の場合に「1」との排他的論理和を取ると結果は1になります。さらに「1」との排他的論理和を取ると結果は「0」になります。つまり「0」→「1」→「0」→……と交互に値を切り換えていくことができます（トグルスイッチ）。

絶対値を求める

ここでは、絶対値を求める方法について解説します。

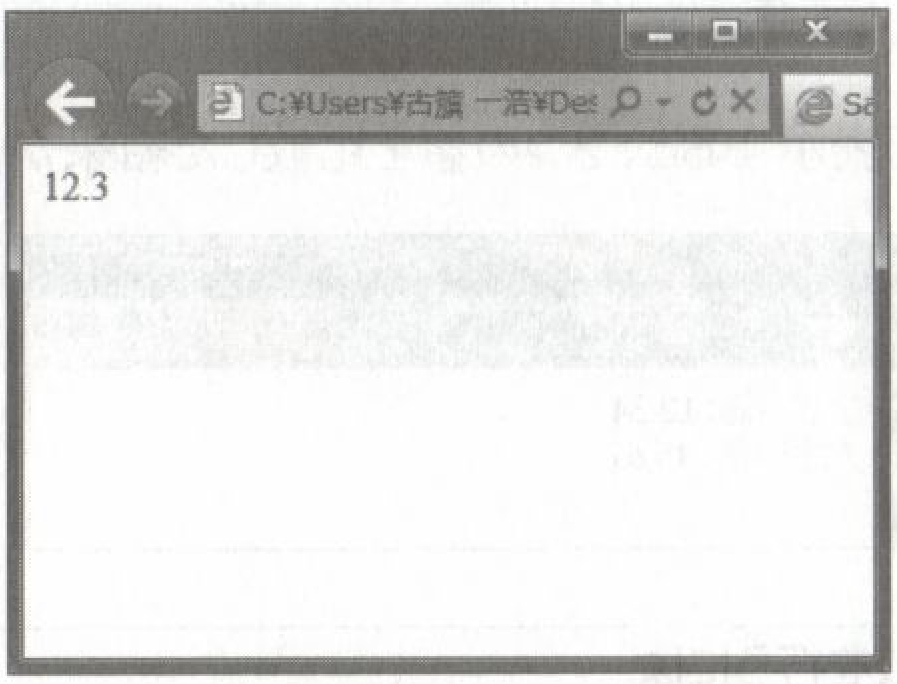
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output>0</output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var n = -12.3;
  // 変数nの絶対値を求める
  var result = Math.abs(n);
  // 結果を出力
  document.getElementsByTagName("output")[0].innerHTML = result;
}, false);
```

ブラウザでindex.htmlを表示すると、「-12.3」の絶対を求めた結果が表示されます。



ONEPOINT 絶対値を求めるには「Math.abs()」メソッドを使う

JavaScriptで絶対を求めるには、「Math.abs()」メソッドを使います。「Math.abs()」メソッドは、値が正数か「0」の場合は、そのままの値を返します。値が負数の場合は正数にして返します。

値を比較する

ここでは、2つの値を比較し、大きい方か小さい方を求める方法について解説します。

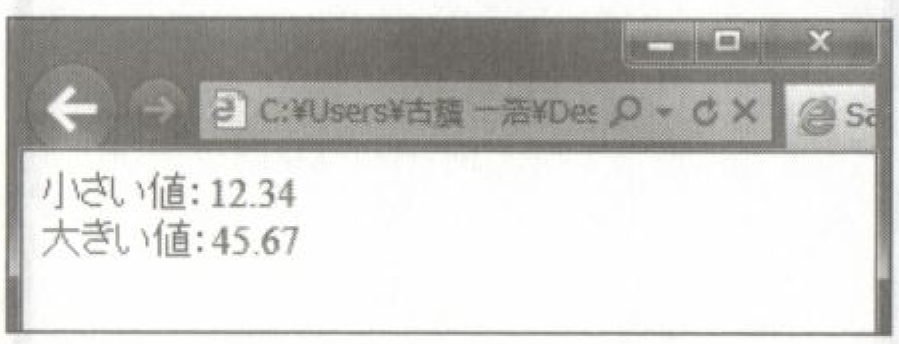
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 比較する値を用意
  var n1 = 12.34;
  var n2 = 45.67;
  // 小さい値を出力
  ele.innerHTML = "小さい値 :"+Math.min(n1, n2)+"<br>";
  // 大きい値を出力
  ele.innerHTML += "大きい値 :"+Math.max(n1, n2)+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、2つの値を比較した結果が表示されます。



ONEPOINT 値の比較を行うには「Math.min()」メソッドや「Math.max()」メソッドを使う

JavaScriptでは、値を比較するためのメソッドが2つ用意されています。2つの値のうち小さい値を求める「Math.min()」メソッドと、大きい値を求める「Math.max()」メソッドです。いずれも2つのパラメータを指定します。「Math.min()」メソッドも「Math.max()」メソッドも比較できるのは2つの値だけで、配列内にある値から大きい値や小さい値を求める場合は「for」文か配列の「forEach()」メソッドなどの高階関数を使う必要があります。

三角関数で計算する

ここでは、三角関数を使った計算の方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

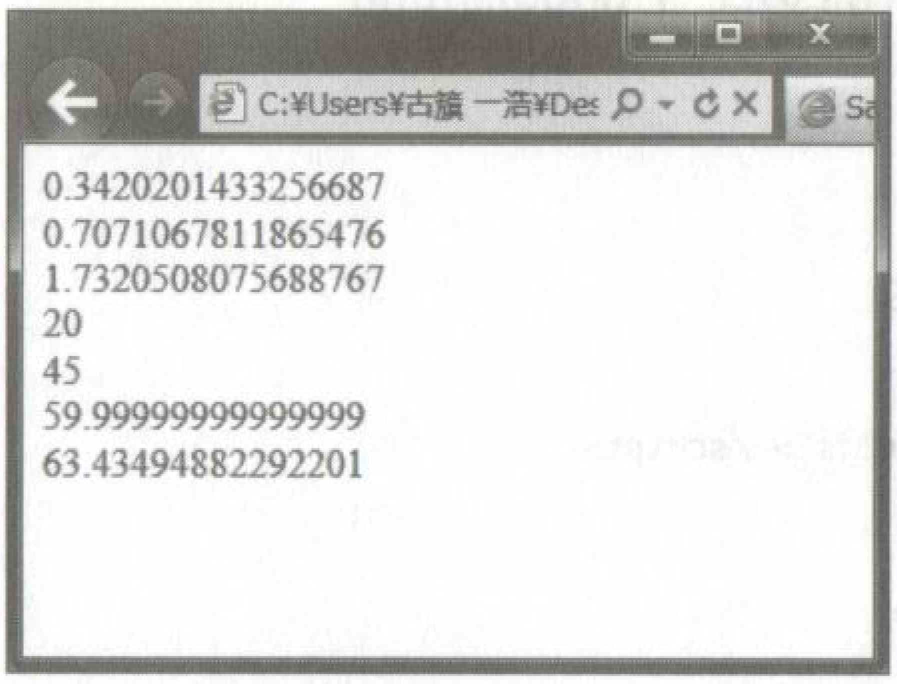
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 20度の正弦値(sin)を求める
  var result1 = Math.sin(20*Math.PI/180);
  ele.innerHTML = result1+"<br>";
  // 45度の余弦値(cos)を求める
  var result2 = Math.cos(45*Math.PI/180);
  ele.innerHTML += result2+"<br>";
  // 60度の正接値(tan)を求める
  var result3 = Math.tan(60*Math.PI/180);
  ele.innerHTML += result3+"<br>";
  // 逆正弦値(asin)を求める(結果は度にして表示)
  var result4 = Math.asin(result1);
  result4 = result4 * 180 / Math.PI;
  ele.innerHTML += result4+"<br>";
  // 逆余弦値(acos)を求める(結果は度にして表示)
  var result5 = Math.acos(result2);
  result5 = result5 * 180 / Math.PI;
  ele.innerHTML += result5+"<br>";
  // 逆正接値(atan)を求める(結果は度にして表示)
  var result6 = Math.atan(result3);
  result6 = result6 * 180 / Math.PI;
  ele.innerHTML += result6+"<br>";
  // x,y座標から角度を求める(結果は度にして表示)
  var x = 50;
  var y = 100;
```

```
    var result7 = Math.atan2(y, x);
    result7 = result7 * 180 / Math.PI;
    ele.innerHTML += result7;
}, false);
```

ブラウザでindex.htmlを表示すると、三角関数を使った計算結果が表示されます。



ONEPOINT **三角関数を取得するには「Math.sin()」メソッドや「Math.cos()」メソッドなどを使う**

JavaScriptでは、次の表に示す三角関数が用意されています。三角関数のメソッドに指定するパラメータは角度(degree)でなく、ラジアン(radian)になります。ラジアンから度に変換するには「ラジアン*180/π」、度からラジアンに変換するは「度*π/180」となります。

なお、演算誤差が発生するため、「Math.sin()」で得た結果を「Math.asin()」のパラメータに指定しても元の値に戻らない点には注意が必要です。

メソッド	説明
sin(n)	正弦/サイン
cos(n)	余弦/コサイン
tan(n)	正接/タンジェント
asin(n)	逆正弦/アークサイン
acos(n)	逆余弦/アークコサイン
atan(n)	逆正接/アークタンジェント
atan2(y, x)	X座標とY座標から角度を求める

円周率を求める

ここでは、JavaScriptでの円周率πについて解説します。

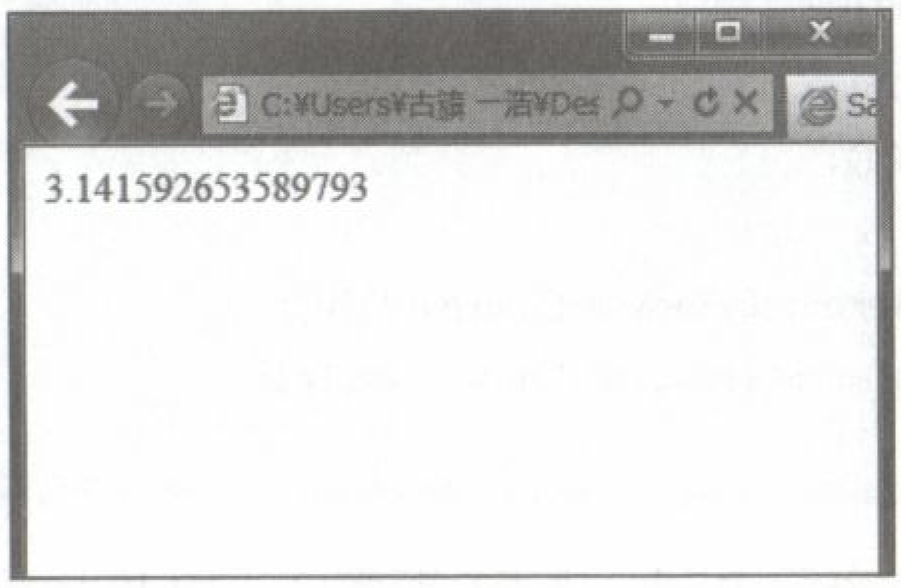
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 円周率を出力
  document.getElementsByTagName("output")[0].innerHTML = Math.PI;
}, false);
```

ブラウザでindex.htmlを表示すると。円周率πの値が表示されます。



ONEPOINT 円周率を扱うには「Math.PI」を使う

JavaScriptでは、πは「Math.PI」として定義されています。「Math.PI」の値は、どのブラウザでも「3.141592653589793」で、小数点以下15桁までです。

COLUMN 度(degree)とラジアン(radian)を変換するには

ラジアンから度に変換するには「ラジアン*180/π」、度からラジアンに変換するは「度*π/180」となります。

四捨五入/切り捨て/切り上げを行う

ここでは、値の小数点以下を四捨五入する方法と、切り捨て、切り上げる方法について解説します。

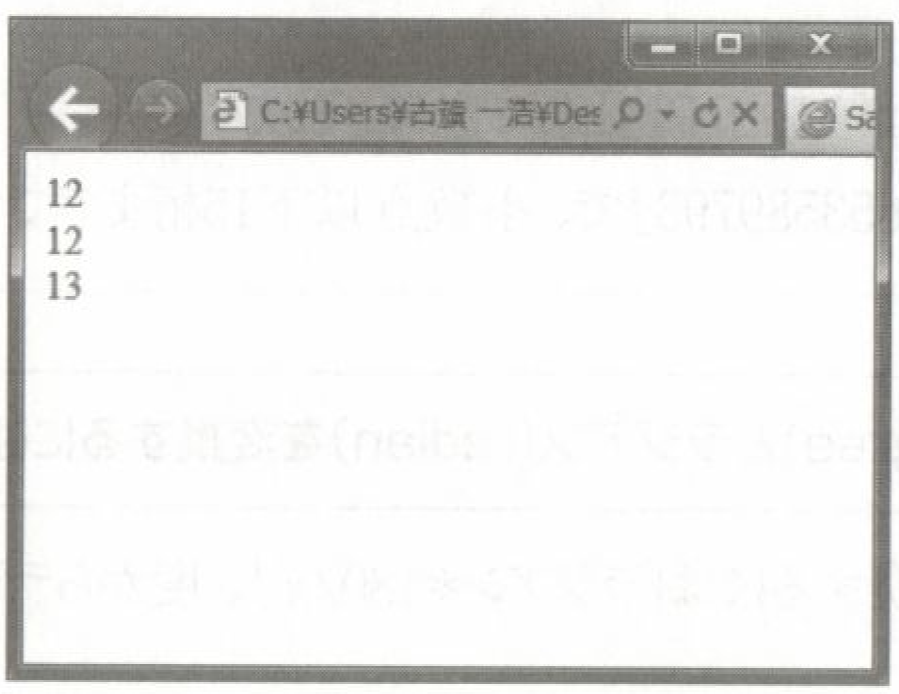
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var n = 12.48;
  // 変数nを四捨五入する
  var result1 = Math.round(n);
  // 変数nの小数値を切り捨てる
  var result2 = Math.floor(n);
  // 変数nの小数値を切り上げる
  var result3 = Math.ceil(n);
  // 結果を出力
  var ele = document.getElementsByTagName("output")[0];
  ele.innerHTML = result1+"<br>"+result2+"<br>"+result3;
}, false);
```

ブラウザでindex.htmlを表示すると、上から順番に、「12.48」の値を四捨五入した値、切り捨てた値、切り上げた値が表示されます。



ONEPOINT 四捨五入を行うには「Math.round()」メソッド、切り捨てを行うには「Math.floor()」メソッド、切り上げを行うには「Math.ceil()」メソッドを使う

JavaScriptでは、小数点以下の処理を行うメソッドが3つ用意されています。四捨五入する場合は「Math.round()」メソッド、切り捨てる場合は「Math.floor()」メソッド、切り上げる場合は「Math.ceil()」メソッドを使います。

COLUMN その他の切り捨て方法

「Math.floor()」による切り捨て以外にも、次の方法があります。

```
数値 | 0
```

```
~~数値
```

上記2つの方法は、数値の範囲によって結果が異なります。「| 0」は32ビット数値範囲を超えると「0」、負数の場合は「1」となります。「~~」の場合は、32ビット数値範囲を超えても正しく切り捨て処理が行われます。ゲームなどで高速化したい場合には32ビット数値範囲を超えないという条件であれば、「| 0」を使うのがほとんどのブラウザを通して高速に処理されます。特にIE9では「Math.floor()」の20〜30倍ほど高速になります。Firefoxでは「Math.floor()」が高速なため、わずかな効果しか得られません。

関連項目 ▶▶▶

● 数値を小数点以下n桁にする ………… p.163

SECTION-072

自然対数の値を求める

ここでは、自然対数の値を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 2~10までの対数を出力
  for(var i=2; i<=10; i++){
    // 対数を求める
    var n = Math.log(i);
    // 結果を出力
    ele.innerHTML += i+" = "+n+"<br>";
  }
  // 自然対数の底eの値を表示
  ele.innerHTML += "自然対数の底e:"+Math.E+"<br>";
  // 2の自然対数値を表示
  ele.innerHTML += "2の自然対数 :"+Math.LN2+"<br>";
  // 10の自然対数値を表示
  ele.innerHTML += "10の自然対数 :"+Math.LN10;
}, false);
```

ブラウザでindex.htmlを表示すると、「2」~「10」までの自然対数の値と、eの値、2の自然対数の値、10の自然対数の値が表示されます。



ONEPOINT

自然対数の値を求めるには「Math.log()」メソッドや「Math.E」「Math.LN2」「Math.LN10」を使う

JavaScriptで自然対数を求めるには、「Math.log()」メソッドを使います。また、使用頻度の高い値に関しては、次のように専用のプロパティが用意されています。

プロパティ	説明
Math.LN2	2の自然対数値
Math.LN10	10の自然対数値
Math.LOG2E	2を底とするeの対数値
Math.LOG10E	10を底とするeの対数値
Math.E	自然対数の底eの値

COLUMN

底が10の場合の対数と底が2の場合の対数を求めるには

底が10の場合の対数と底が2の場合の対数を求めるには、次のようになります。

$\log_{10}x$ = Math.LOG10E * Math.log(x)
$\log_{2}x$ = Math.LOG2E * Math.log(x)

具体的なJavaScriptのコードは、次のようになります。

```
// 底が10の対数を求める
var x  = 1000;
var log10x  = Math.LOG10E * Math.log(x);
// 10の自然対数値を表示
ele.innerHTML = log10x + "<br>";
// 底が2の対数を求める
var x  = 65536;
var log2x  = Math.LOG2E * Math.log(x);
```

関連項目 ▶▶▶

● eの累乗を求める ………………………………………… p.188

eの累乗を求める

ここでは、JavaScriptでの自然対数の底eの累乗計算について解説します。

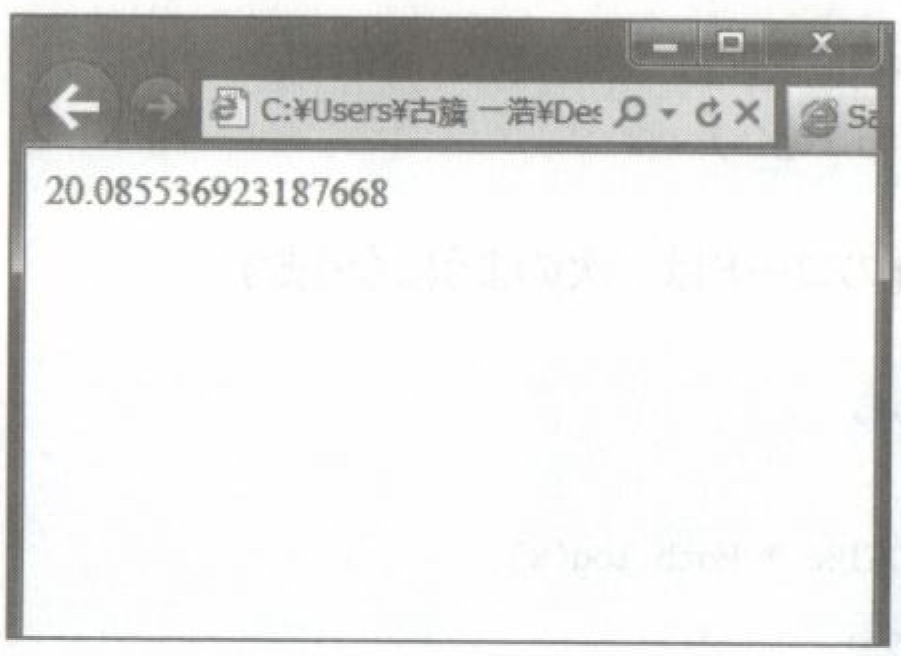
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // eの累乗を出力
  document.getElementsByTagName("output")[0].innerHTML = Math.exp(3);
}, false);
```

ブラウザでindex.htmlを表示すると、自然対数の底eを累乗(サンプルでは3乗)した値が表示されます。



ONEPOINT eの累乗を求めるには「Math.exp()」メソッドを使う

JavaScriptでは自然対数の底の値であるeの累乗は「Math.exp()」メソッドで計算することができます。「Math.exp()」メソッドのパラメータは1つで、「1」を指定すると「e^1」、「7」であれば「e^7」と同じになります。

関連項目 ▶▶▶

● 自然対数の値を求める……………………………………p.186

べき乗を求める

ここでは、べき乗(xのn乗)を求める方法について解説します。

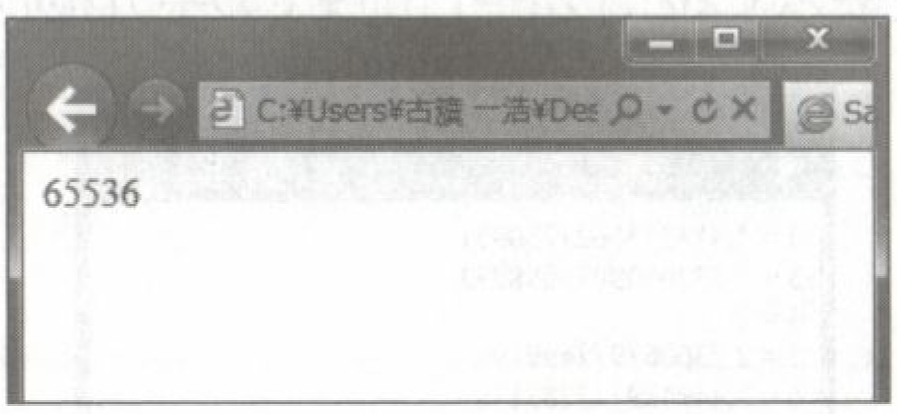
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 2の16乗を求める
  var n = Math.pow(2, 16);
  // 値を出力
  document.getElementsByTagName("output")[0].innerHTML = n;
}, false);
```

ブラウザでindex.htmlを表示すると、2の16乗を計算した結果が表示されます。



ONEPOINT べき乗計算を行うには「Math.pow()」メソッドを使う

JavaScriptでべき乗(xのn乗)を計算するには、「Math.pow()」メソッドを使います。2つ目のパラメータには、整数値以外も指定することができます。たとえば、1/2を指定するとxの平方根、1/3を指定するとxの3乗根となります。また、負数を指定した場合はxの-n乗になります。計算結果が範囲を超えている場合は、「Infinity」、または、「-Infinity」を返します。

関連項目 ▶▶▶

● 平方根を求める ……… p.190

平方根を求める

ここでは、平方根を求める方法について解説します。

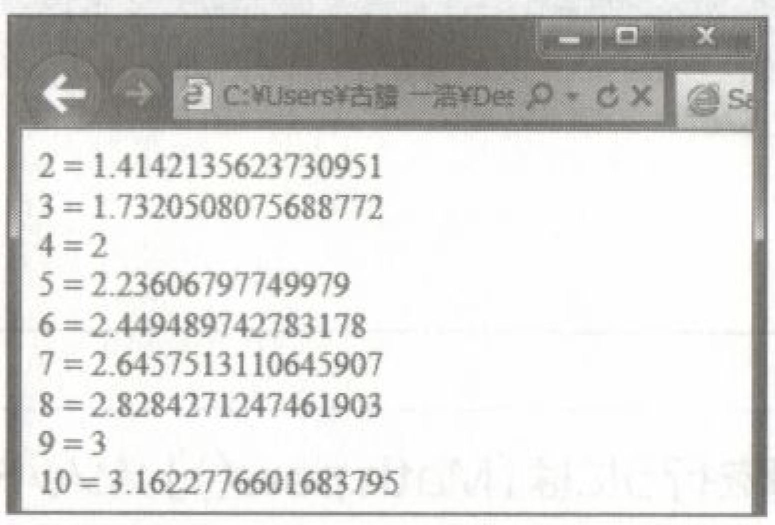
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 2～10までの平方根の値を求める
  for(var i=2; i<=10; i++){
    var n = Math.sqrt(i);  // 平方根を求める
    document.getElementsByTagName("output")[0].innerHTML += i+" = "+n+"<br>";  // 結果を出力
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「2」～「10」までの平方根の値が表示されます。



ONEPOINT 平方根を求めるには「Math.sqrt()」メソッドを使う

JavaScriptで平方根を求めるには、「Math.sqrt()」メソッドを使います。パラメータには平方根を求めたい値を指定します。また、2の平方根は「Math.SQRT2」、2の平方根の半分の値については「Math.SQRT1_2」として「Math」オブジェクトに用意されています。

関連項目 ▶▶▶

● べき乗を求める…………p.189

乱数を求める

ここでは、乱数を求める方法について解説します。

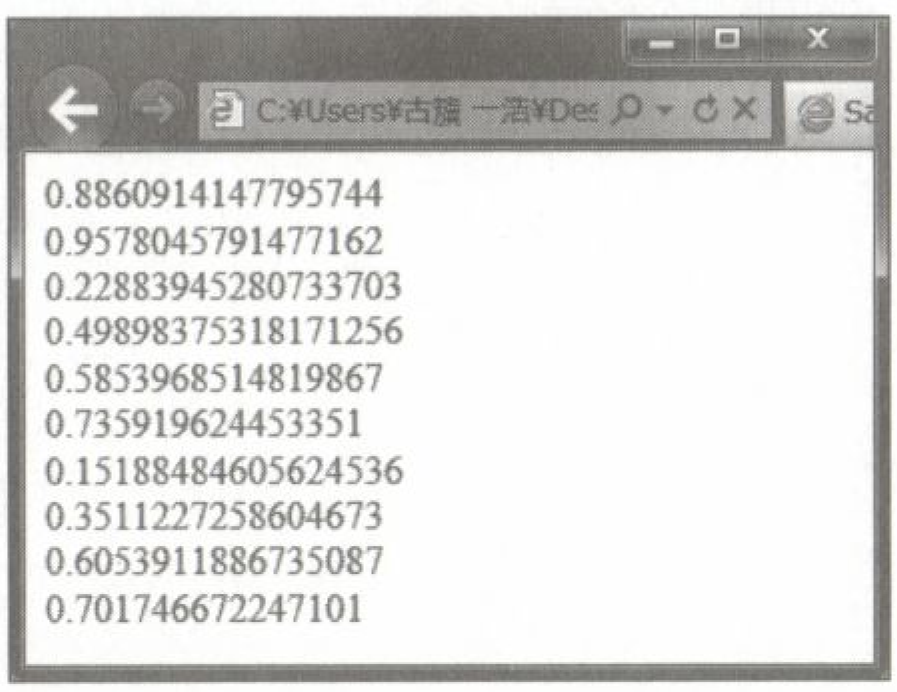
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 乱数を10回発生させる
  for(var i=0; i<10; i++){
    // 乱数値を求める
    var n = Math.random();
    // 結果を出力
    document.getElementsByTagName("output")[0].innerHTML += n+"<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、10個の乱数値が表示されます。



ONEPOINT 乱数を求めるには「Math.random()」メソッドを使う

JavaScriptで乱数を発生させるには、「Math.random()」メソッドを使います。パラメータはありません。「Math.random()」メソッドが返す値は、0以上1未満の小数値になります。発生する乱数は疑似乱数となっています。また、発生される乱数の種(seed)を指定することはできません。

COLUMN 整数値の乱数を求めるには

「Math.random()」メソッドは0以上1未満の小数値を返しますが、場合によっては整数の乱数を得たいこともあります。

この場合、32ビット数値範囲内であれば、次のように、「求めたい乱数値の値+1」を乗算した後、「| 0」とすることで整数の乱数を求めることができます。次の場合は、0~9までの整数の乱数値を求めます。

```
Math.random() * 10 | 0
```

1~10までの乱数値であれば、次のようになります。

```
(Math.random() * 10 | 0) + 1
```

なお、32ビット範囲を超える場合は「| 0」ではなく、「Math.floor()」メソッドか「Math.ceil()」メソッド、もしくは、「~~」を使用します。

```
Math.floor(Math.random() * 10)
Math.ceil(Math.random() * 10)
~~(Math.random() * 10)
```

CHAPTER 05

文字列・正規表現

「String」オブジェクトを生成する/文字列を連結する

ここでは、「String」オブジェクトを生成する方法と文字列同士を連結するについて解説します。

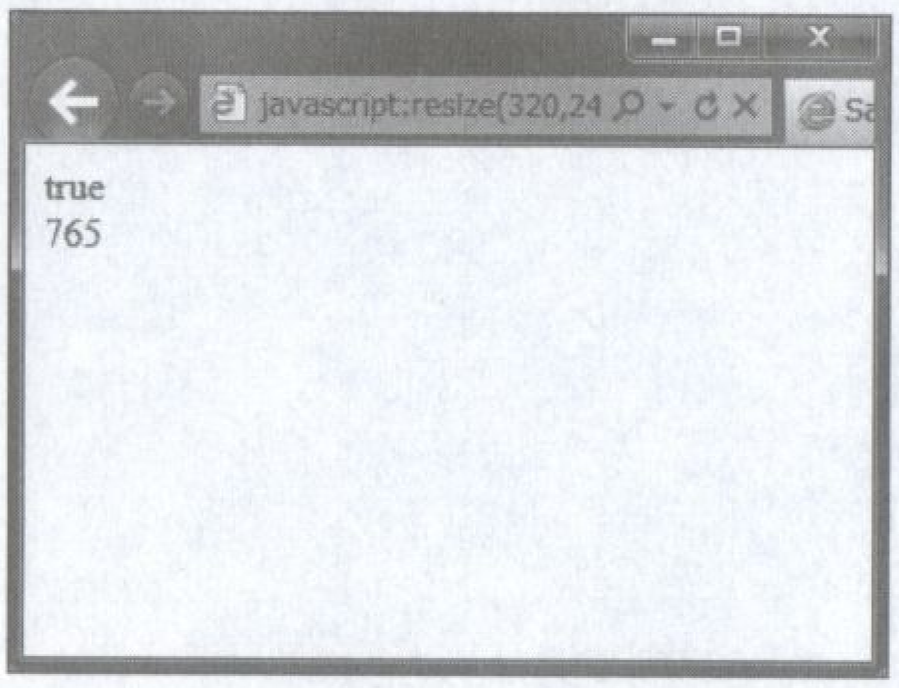
SAMPLE CODE HTML(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Stringオブジェクトを生成
  var str1 = new String();
  // 文字列として76を生成し、5を連結
  var str2 = new String(76);
  str2 = str2 + 5;
  // 結果を出力
  ele.innerHTML = (str1 instanceof String)+"<br>"+str2;
}, false);
```

ブラウザでindex.htmlを表示すると、「String」オブジェクトの生成結果が表示されます。2行目は、文字列と数値を連結した結果が表示されます。



ONEPOINT 「String」オブジェクトを生成するには「new String()」や「"」、「'」を使う

JavaScriptでは、「String」オブジェクトを使って文字型のオブジェクトを生成することができます。文字列型のオブジェクトを生成するには「new String()」とします。「String」オブジェクトを生成する際にパラメータを指定すると、指定されたパラメータが文字列として生成されます。また、「new String()」として生成しなくても「"」(ダブルクォーテーション)、または、「'」(シングルクォーテーション)を使って、数値や文字を囲むことで文字列として扱うことができます。

なお、変数の型を調べる際に「typeof」演算子を使うと、文字列を格納した変数やオブジェクトは「object」を返します。サンプルのように「instanceof」演算子を使えば、生成元のオブジェクトを示す「String」の文字が返されます。

文字列同士を連結する場合には「+」記号を使います。文字列と数値を連結した場合には、数値が文字に変換され連結されます。たとえば、「"ab"+3」とすると、結果は「ab3」になります。「"12"+3」も文字列と数値の連結になるので「15」にならず、「123」の文字列になります。

COLUMN サロゲートペアの文字

サロゲートペアを利用している文字はJavaScriptでは表示できなかったり、文字数が正しくカウントされません。サロゲートペアを利用した文字を表示するには、次のように処理します。

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var c1 = 0x20213;
  // 結果を出力
  ele.innerHTML = uniConv(c1);
}, false);
// Unicode サロゲートペア文字を表示可能な文字に変換
function uniConv(x){
  x = x - 0x10000;
  return String.fromCharCode(((x / 0x400) | 0) + 0xD800, (x % 0x400) + 0xDC00);
}
```

JavaScriptで文字列処理を行う場合、サロゲートペアを利用した文字が含まれる可能性がある場合は、既存のメソッドやプロパティは期待通りの動作ではなくなるため、別途、処理する必要があります。サロゲートペアを利用している文字の処理方法については、次のサイトを参考にしてください。

● 参考サイト

URL http://d.hatena.ne.jp/language_and_engineering/20120523/LengthOfUnicodeStringWithSurrogatePairs

URL http://codezine.jp/article/detail/1592

URL http://liosk.blog103.fc2.com/blog-entry-162.html

なお、サロゲートペアを利用している文字は、次のようになります。

𠀋𡈽𡌛𡑮𡢽𠮟𡚴𡸴𣇄𣗄𣜿𣝣𣳾𤟱𥒎𥔎𥝱𥧄𥶡𦫿𦹀𧃴𧚄𨉷𨏍𪆐𠂉𠂢𠂤𠆢𠈓𠌫𠎁𠍱𠏹𠑊𠔉𠗖𠘨𠝏𠠇𠠺𠢹𠥼𠦝𠫓𠬝𠵅𠷡𠺕𠹭𠹤𠽟𡈁𡉕𡉻𡉴𡋤𡋗𡋽𡌶𡍄𡏄𡑭𡗗𦰩𡙇𡜆𡝂𡧃𡱖𡴭𡵅𡵸𡵢𡶡𡶜𡶒𡶷𡷠𡸳𡼞𡽶𡿺𢅻𢌞𢎭𢛳𢡛𢢫𢦏𢪸𢭏𢭐𢭆𢰝𢮦𢰤𢷡𣇃𣇵𣆶𣍲𣏓𣏒𣏐𣏤𣏕𣏚𣏟𣑊𣑑𣑋𣑥𣓤𣕚𣖔𣘹𣙇𣘸𣘺𣜜𣜌𣝤𣟿𣟧𣠤𣠽𣪘𣱿𣴀𣵀𣷺𣷹𣷓𣽾𤂖𤄃𤇆𤇾𤎼𤘩𤚥𤢖𤩍𤭖𤭯𤰖𤴔𤸎𤸷𤹪𤺋𥁊𥁕𥄢𥆩𥇥𥇍𥈞𥉌𥐮𥓙𥖧𥞩𥞴𥧔𥫤𥫣𥫱𥮲𥱋𥱤𥸮𥹖𥹥𥹢𥻘𥻂𥻨𥼣𥽜𥿠𥿔𦀌𥿻𦀗𦁠𦃭𦉰𦊆𦍌𣴎𦐂𦙾𦚰𦜝𦣝𦣪𦥑𦥯𦧝𦨞𦩘𦪌𦪷𦱳𦳝𦹥𦾔𦿸𦿶𦿷𧄍𧄹𧏛𧏚𧏾𧐐𧑉𧘕𧘔𧘱𧚓𧜎𧜣𧝒𧦅𧪄𧮳𧮾𧯇𧲸𧶠𧸐𧾷𨂊𨂻𨊂𨋳𨐌𨑕𨕫𨗈𨗉𨛗𨛺𨥉𨥆𨥫𨦇𨦈𨦺𨦻𨨞𨨩𨩱𨩃𨪙𨫍𨫤𨫝𨯁𨯯𨴐𨵱𨷻𨸟𨸶𨺉𨻫𨼲𨿸𩊠𩊱𩒐𩗏𩙿𩛰𩜙𩝐𩣆𩩲𩷛𩸽𩸕𩺊𩹉𩻄𩻩𩻛𩿎𪀯𪀚𪃹𪂂𢈘𪎌𪐷𪗱𪘂𪘚𪚲

COLUMN Latin-1コード表

「charAt()」メソッドや「charCodeAt()」メソッド、「String.fromCharCode()」メソッドなどで使用されるLatin-1コード表は、次のようになります。

上位ビット / 下位ビット

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td>0</td><td>@</td><td>P</td><td>`</td><td>p</td><td></td><td></td><td></td><td>°</td><td>À</td><td>Ð</td><td>à</td><td>ð</td></tr>
<tr><td>1</td><td></td><td></td><td>!</td><td>1</td><td>A</td><td>Q</td><td>a</td><td>q</td><td></td><td></td><td>¡</td><td>±</td><td>Á</td><td>Ñ</td><td>á</td><td>ñ</td></tr>
<tr><td>2</td><td></td><td></td><td>"</td><td>2</td><td>B</td><td>R</td><td>b</td><td>r</td><td></td><td></td><td>¢</td><td>²</td><td>Â</td><td>Ò</td><td>â</td><td>ò</td></tr>
<tr><td>3</td><td></td><td></td><td>#</td><td>3</td><td>C</td><td>S</td><td>c</td><td>s</td><td></td><td></td><td>£</td><td>³</td><td>Ã</td><td>Ó</td><td>ã</td><td>ó</td></tr>
<tr><td>4</td><td></td><td></td><td>$</td><td>4</td><td>D</td><td>T</td><td>d</td><td>t</td><td></td><td></td><td>¤</td><td>´</td><td>Ä</td><td>Ô</td><td>ä</td><td>õ</td></tr>
<tr><td>5</td><td></td><td></td><td>%</td><td>5</td><td>E</td><td>U</td><td>e</td><td>u</td><td></td><td></td><td>¥</td><td>µ</td><td>Å</td><td>Õ</td><td>å</td><td>ô</td></tr>
<tr><td>6</td><td></td><td></td><td>&</td><td>6</td><td>F</td><td>V</td><td>f</td><td>v</td><td></td><td></td><td>¦</td><td>¶</td><td>Æ</td><td>Ö</td><td>æ</td><td>ö</td></tr>
<tr><td>7</td><td></td><td></td><td>'</td><td>7</td><td>G</td><td>W</td><td>g</td><td>w</td><td></td><td></td><td>§</td><td>·</td><td>Ç</td><td>×</td><td>ç</td><td>÷</td></tr>
<tr><td>8</td><td></td><td></td><td>(</td><td>8</td><td>H</td><td>X</td><td>h</td><td>x</td><td></td><td></td><td>¨</td><td>¸</td><td>È</td><td>Ø</td><td>è</td><td>ø</td></tr>
<tr><td>9</td><td></td><td></td><td>)</td><td>9</td><td>I</td><td>Y</td><td>i</td><td>y</td><td></td><td></td><td>©</td><td>¹</td><td>É</td><td>Ù</td><td>é</td><td>ù</td></tr>
<tr><td>A</td><td></td><td></td><td>*</td><td>:</td><td>J</td><td>Z</td><td>j</td><td>z</td><td></td><td></td><td>ª</td><td>º</td><td>Ê</td><td>Ú</td><td>ê</td><td>ú</td></tr>
<tr><td>B</td><td></td><td></td><td>+</td><td>;</td><td>K</td><td>[</td><td>k</td><td>{</td><td></td><td></td><td>«</td><td>»</td><td>Ë</td><td>Û</td><td>ë</td><td>û</td></tr>
<tr><td>C</td><td></td><td></td><td>,</td><td><</td><td>L</td><td>\</td><td>l</td><td>|</td><td></td><td></td><td>¬</td><td>¼</td><td>Ì</td><td>Ü</td><td>ì</td><td>ü</td></tr>
<tr><td>D</td><td></td><td></td><td>-</td><td>=</td><td>M</td><td>]</td><td>m</td><td>}</td><td></td><td></td><td>`</td><td>½</td><td>Í</td><td>Ý</td><td>í</td><td>ý</td></tr>
<tr><td>E</td><td></td><td></td><td>.</td><td>></td><td>N</td><td>^</td><td>n</td><td>~</td><td></td><td></td><td>®</td><td>¾</td><td>Î</td><td>Þ</td><td>î</td><td>þ</td></tr>
<tr><td>F</td><td></td><td></td><td>/</td><td>?</td><td>O</td><td>_</td><td>o</td><td></td><td></td><td></td><td>¯</td><td>¿</td><td>Ï</td><td>ß</td><td>ï</td><td>ÿ</td></tr>
</table>

文字列の長さを求める

ここでは、文字列の長さを求める方法について解説します。

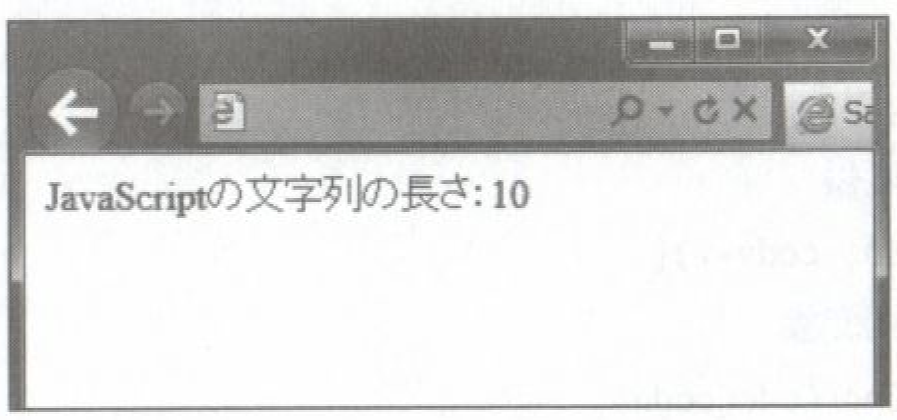
SAMPLE CODE **HTMLのコード(index.html)**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE **JavaScriptのコード(sample.js)**

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var str = "JavaScript";
  // 文字列の長さを求める
  var n = str.length;
  // 結果を出力
  ele.innerHTML = str+"の文字列の長さ:"+n;
}, false);
```

ブラウザでindex.htmlを表示すると、文字列の長さが表示されます。



ONEPOINT 文字列の長さを求めるには「length」プロパティを使う

文字列の長さを求めるには、「length」プロパティを参照します。文字は日本語でも中国語でもアラビア語でも1文字を「1」としてカウントされます。

なお、サロゲートペアを利用した文字の場合、1文字でも「2」が返されます。文字列にサロゲートペアを利用した文字が含まれる場合には別途、処理が必要になります。詳しくは、195ページを参照してください。

SECTION-079

文字と文字コードを変換する

ここでは、文字から文字コードを求める方法と、文字コードから文字を生成する方法について解説します。

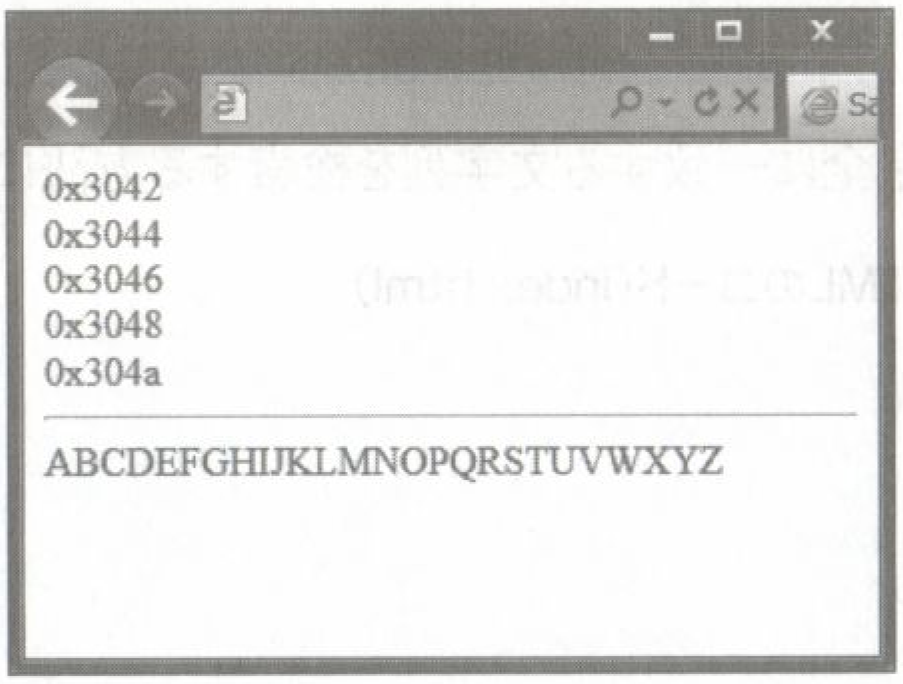
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var str = "あいうえお";
  // 文字数だけ繰り返し変換
  for(var i=0; i<str.length; i++){
    // 文字コードに変換
    var code = "0x"+str.charCodeAt(i).toString(16);
    // 結果を出力
    ele.innerHTML += code+"<br>";
  }
  ele.innerHTML += "<hr>";
  // 文字コードから文字に変換
  for(code=0x41; code<=0x5a; code++){
    // 文字コードから文字に変換
    var str = String.fromCharCode(code);
    // 結果を出力
    ele.innerHTML += str;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、文字コードに変換した結果と、文字コード順に文字を表示した結果が表示されます。



ONEPOINT **文字を文字コードに変換するには「charCodeAt()」メソッド、文字コードを文字に変換するには「String.fromCharCode()」メソッドを使う**

文字から文字コードに変換するには、「charCodeAt()」メソッドを使います。パラメータには変換したい文字の位置を指定します。「0」が最初の文字を示し、以後、1ずつ増加します。

文字コードから文字に変換するには、「String.fromCharCode()」メソッドを使います。パラメータには文字コードを指定します。「String.fromCharCode()」メソッドは変換したい文字コードの数だけ、「,」(カンマ)で区切って列記することで一括して文字コードから文字に変換することができます。

COLUMN **文字コードについて**

コンピュータで仕様される文字には「文字コード」が割り当てられています。文字コードは複数の種類があり、2012年現在、Webで多く利用されているのはASCIIかUTF-8です。他にも日本語の場合Shitf JISやEUCがあります。主な記号と英数字はどの文字コードでも共通していますが、それ以外の部分はHTMLファイルに依存します。同じ文字であってもUTF-8の文字コードで保存されている場合、UTF-8でのコードを返します。Shift JISであれば、Shift JISでの文字コードを返すことになります。

文字列を検索する

ここでは、文字列から完全に一致する文字列を検索する方法について解説します。

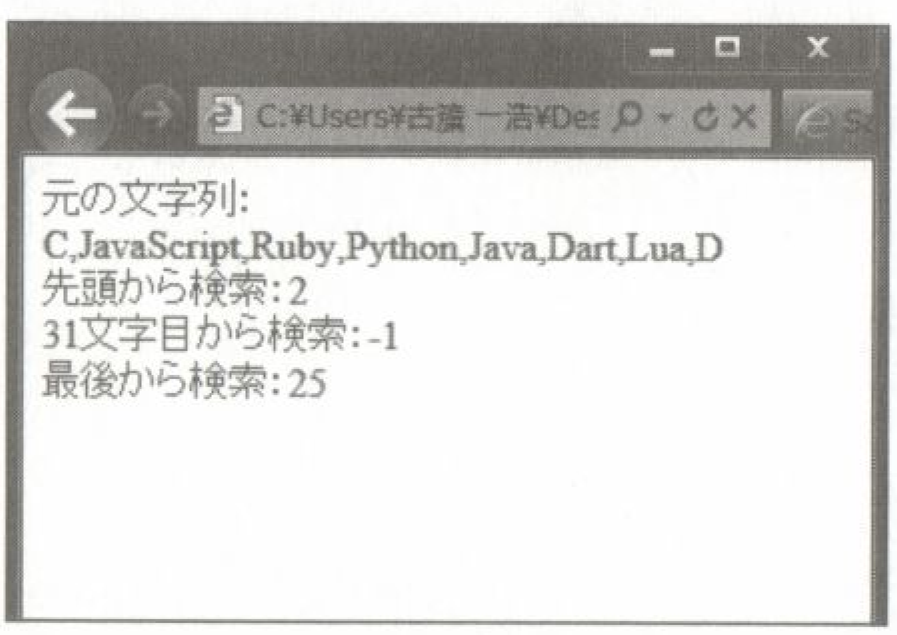
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var myString = "C,JavaScript,Ruby,Python,Java,Dart,Lua,D";
  // 先頭からJavaの文字を探す
  var ptr1 = myString.indexOf("Java");
  // 31文字目からJavaの文字を探す
  var ptr2 = myString.indexOf("Java", 30);
  // 文字列の最後からJavaの文字を探す
  var ptr3 = myString.lastIndexOf("Java");
  // 結果を出力
  ele.innerHTML = "元の文字列 :"+myString+"<br>";
  ele.innerHTML += "先頭から検索 :"+ptr1+"<br>31文字目から検索 :"+ptr2+"<br>";
  ele.innerHTML += "最後から検索 :"+ptr3;
}, false);
```

ブラウザでindex.htmlを表示すると、文字列から「Java」の文字を検索した結果が表示されます。



ONEPOINT

文字列の検索を行うには「indexOf()」メソッドや「lastIndexOf()」メソッドを使う

JavaScriptでは文字列の検索を行うには、「indexOf()」メソッドや「lastIndexOf()」メソッドを使います。

「indexOf()」メソッドは文字列の先頭から末尾に向かって検索し、「lastIndexOf()」メソッドは文字列の末尾から先頭に向かって検索します。該当する文字列が見つかった場合は、文字列内での先頭からの位置を返します（最初の文字位置は「0」）。文字列が見つからなかった場合には「-1」を返します。また、「indexOf()」メソッド、「lastIndexOf()」メソッドとも、文字列の検索開始位置を2番目のパラメータで指定することができます。2番目のパラメータを省略した場合、「indexOf()」メソッドは文字列の先頭から、「lastIndexOf()」メソッドは文字列の末尾から検索するように設定されます。

なお、「indexOf()」メソッドと「lastIndexOf()」メソッドは、シリアルサーチで、文字列の先頭、または、末尾から順番に指定された文字列があるかを調べます。より複雑な検索を行う場合には、正規表現を利用できる「match()」メソッドを使います。「match()」メソッドに関しては、216ページを参照してください。単純に文字列を検索する場合は、シリアルサーチを行う「indexOf()」メソッドや「lastIndexOf()」メソッドの方が高速です。

関連項目 ▶▶▶

- 配列要素を検索する ……… p.134
- 正規表現を使って文字を検索する ……… p.216

文字を取り出す

ここでは、文字列から指定した位置にある文字を取り出す方法について解説します。

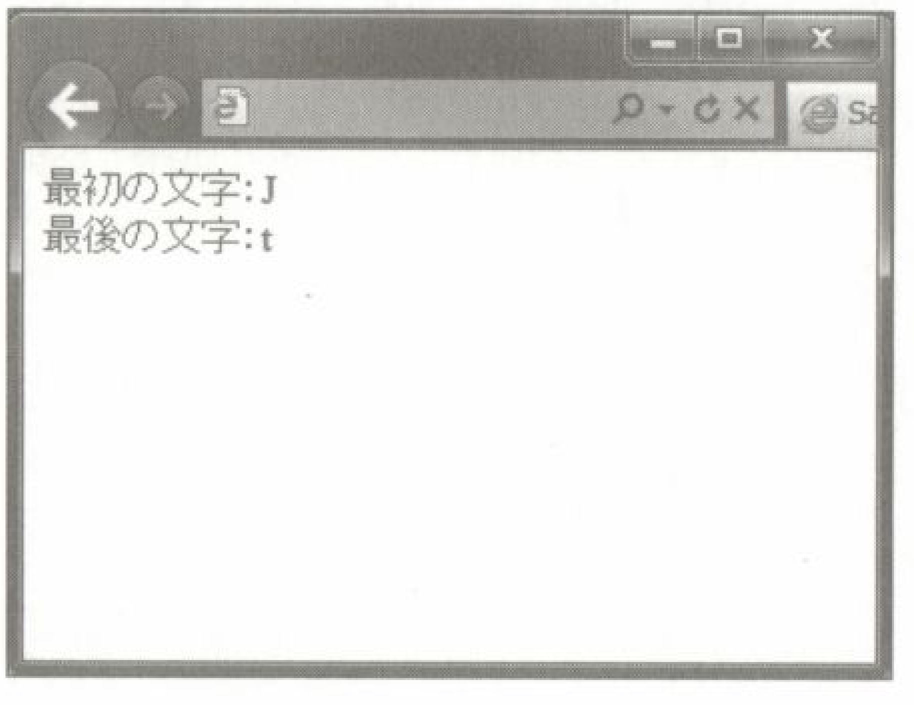
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var myString = "JavaScript";
  // 最初の文字を取り出す
  var chr1 = myString.charAt(0);
  // 最後の文字を取り出す
  var chr2 = myString.charAt(myString.length-1);
  // 結果を出力
  ele.innerHTML = "最初の文字 :"+chr1+"<br>最後の文字 :"+chr2;
}, false);
```

ブラウザでindex.htmlを表示すると、文字列の最初と最後の文字が表示されます。



ONEPOINT 文字を取り出すには「charAt()」メソッドを使う

文字列内から1文字を取り出すには、「charAt()」メソッドを使います。パラメータには取り出す文字の位置を指定します。この文字の位置は、最初の文字が「0」、その次の文字が「1」となり、以後、1ずつ増加していきます。つまり、最初の文字を取り出す場合は「charAt(0)」、2番目の文字を取り出す場合は「charAt(1)」となります。なお、最後の文字を取り出すには、サンプルのように、文字列の長さから1減らした値を指定します。「charAt(-1)」のように指定しても最後の文字を取り出すことはできません。

複数の文字を取り出す場合は、「substr()」メソッドや「substring()」メソッドを使います。また、正規表現を利用してマッチする文字を取り出す方法もあります。

COLUMN 文字を文字コードで取得するには

文字列から文字ではなく、文字コードで取得したい場合もあります。文字コードで取得する場合には、「charCodeAt()」メソッドを使用します。

たとえば、次のようにすると、「a」の文字コードである「97」が返されます。

```
"a".charCodeAt(0)
```

関連項目 ▶▶▶

● 指定範囲の文字を抜き出す ………… p.206
● 正規表現を使って文字を検索する ………… p.216

SECTION-082

文字列を指定文字で分割する

ここでは、文字列を指定文字で分割する方法について解説します。

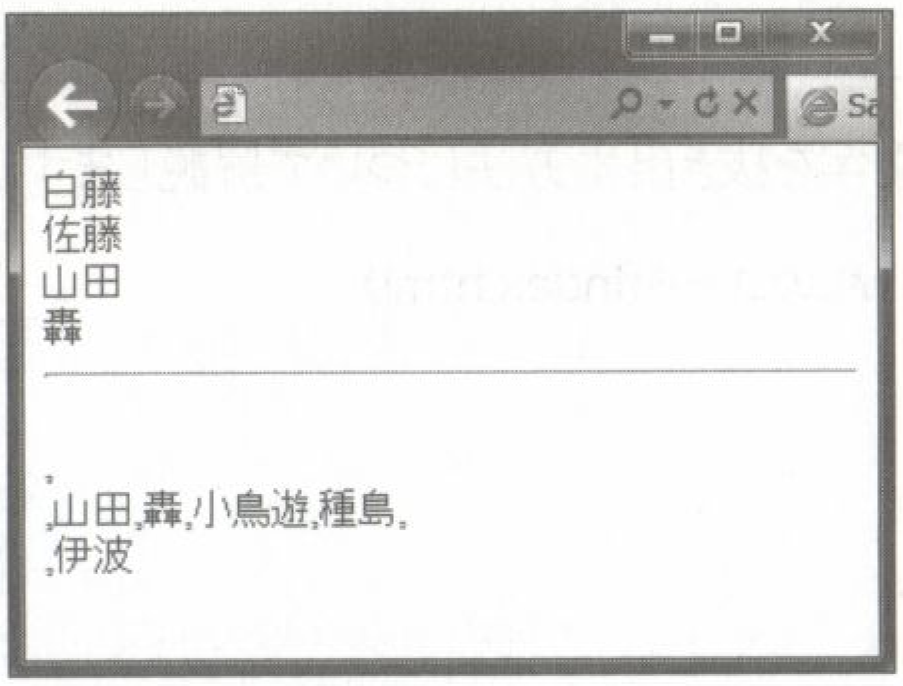
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var myString = "白藤,佐藤,山田,轟,小鳥遊,種島,大藤,伊波";
  // ,の文字で分割
  var myArray = myString.split(",", 4);
  // 結果を出力
  for(var i=0; i<myArray.length; i++){
    var str = myArray[i];
    ele.innerHTML += str+"<br>";
  }
  ele.innerHTML += "<hr>";
  // 正規表現を使って「●藤」,の文字を削除
  var myArray2 = myString.split(/.藤/g);
  // 結果を出力
  for(var i=0; i<myArray2.length; i++){
    var str = myArray2[i];
    ele.innerHTML += str+"<br>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、文字列を特定文字、および、正規表現で分割した結果が表示されます。



ONEPOINT **文字列を指定文字で分割するには「split()」メソッドを使う**

文字列内にある特定の文字を区切り文字として分割し配列に格納するには、「split()」メソッドを使います。「split()」メソッドの最初のパラメータには、区切り文字を指定します。分割する際、大量に配列データが生成されますが、個数を制限したい場合には2番目のパラメータに最大数を指定します。たとえば、「split("■", 3)」であれば、「■」を区切り文字として最大3つまでの結果が返されます。

また、固定された文字列ではなく、柔軟性を持たせたい場合には、正規表現文字列を「split()」の最初のパラメータに指定することができます。

関連項目 ▶▶▶

● 特定の文字列を削除する …… p.211
● 「RegExp」オブジェクトを生成する …… p.213

SECTION-083

指定範囲の文字を抜き出す

ここでは、指定範囲の文字を抜き出す方法について解説します。

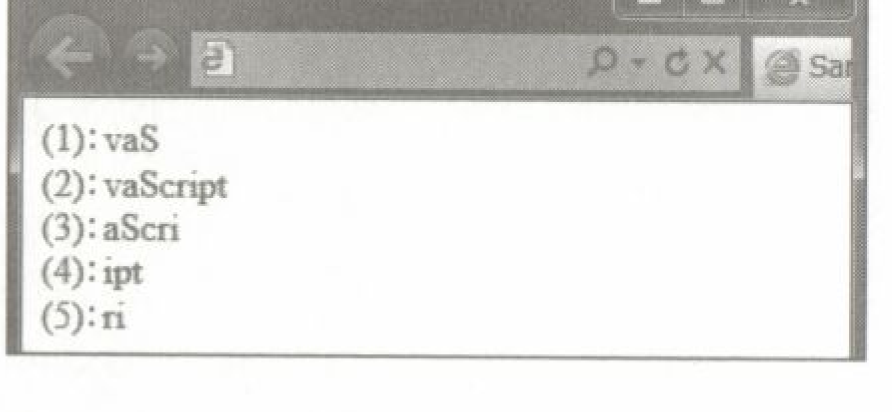
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var myString = "JavaScript";
  // 3文字目～5文字目の文字を取り出す
  var str1 = myString.substring(2, 5);
  // 3文字目から最後まで取り出す
  var str2 = myString.substring(2);
  // 4文字目から5文字分を取り出す
  var str3 = myString.substr(3, 5);
  // 文字列の最後から3文字分を取り出す
  var str4 = myString.substr(-3);
  // 文字列の最後から数えて2～4文字分を取り出す
  var str5 = myString.slice(-4, -2);
  // 結果を出力
  ele.innerHTML = "(1):"+str1+"<br>(2):"+str2+"<br>";
  ele.innerHTML += "(3):"+str3+"<br>(4):"+str4+"<br>(5):"+str5;
}, false);
```

ブラウザでindex.htmlを表示すると、同じ文字列から「substring()」メソッドと「substr()」メソッドを使って抜き出した結果が表示されます。

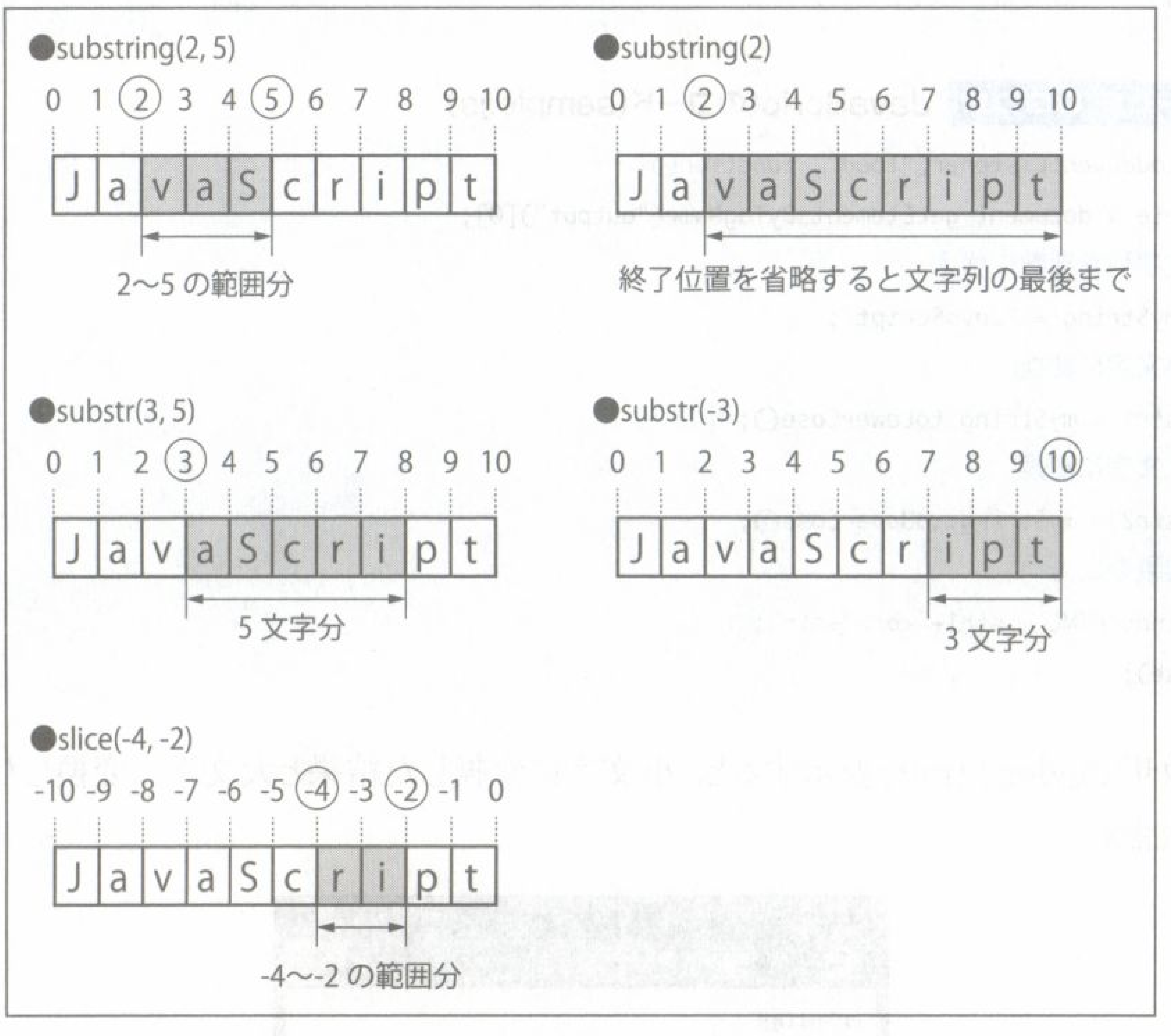


ONEPOINT 文字列を抜き出すには「substring()」メソッドや「slice()」メソッド、「substr()」メソッドを使う

指定範囲の文字列を抜き出すには、「substring()」メソッドや「slice()」メソッド、「substr()」メソッドを使います。文字の範囲を指定して抜き出す場合は「subtring()」メソッドか「slice()」メソッド、文字数を指定して抜き出す場合は「substr()」メソッドを使います。文字の位置は、下図に示すように、文字を挟むようになっており、パラメータには、この範囲を指定することになります。たとえば、「substring(2, 5)」とすると、下図のように、3文字目から5文字目までが抜き出されます。

「substring()」メソッドは、負数を指定して文字列の末尾から抜き出すことはできません。負数を指定して末尾から抜き出すには、「slice()」メソッドを使います。「substring()」メソッドも「slice()」メソッドも、最初のパラメータが開始位置、2番目が終了位置になります。

「substr()」メソッドは、最初に指定したパラメータが示す文字列の位置から、2番目に指定したパラメータ分だけ文字列を抜き出します。また、最初のパラメータに負数を指定すると、末尾から抜き出すことができます。



関連項目 ▶▶▶

● 文字を取り出す …………………………………………………… p.202

英文字の大文字/小文字に変換する

ここでは、英文字の大文字/小文字に変換する方法について解説します。

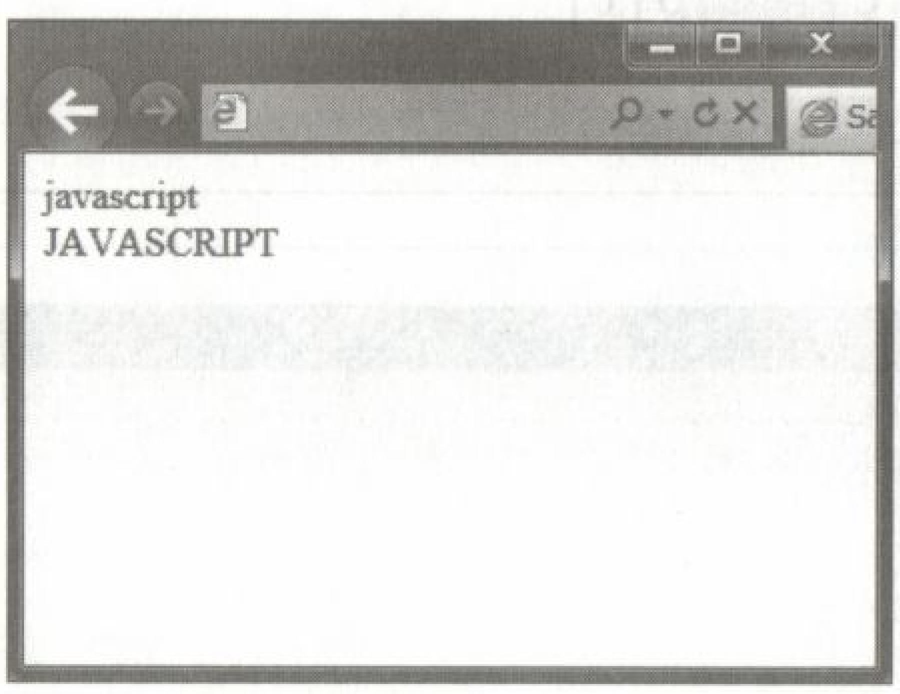
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入
  var myString = "JavaScript";
  // 小文字に変換
  var str1 = myString.toLowerCase();
  // 大文字に変換
  var str2 = myString.toUpperCase();
  // 結果を出力
  ele.innerHTML = str1+"<br>"+str2;
}, false);
```

ブラウザでindex.htmlを表示すると、小文字に変換した結果と大文字に変換した結果が表示されます。



ONEPOINT

英文字を小文字に変換するには「toLowerCase()」メソッド、大文字に変換するには「toUpperCase()」メソッドを使う

英文字は、大文字と小文字に変換することができます。小文字に変換する場合は、「toLowerCase()」メソッドを使います。大文字に変換する場合は、「toUpperCase()」メソッドを使います。

なお、小文字を小さい大文字(スモールキャップス)にするメソッドは、JavaScriptには用意されていません。スモールキャップスを表現したい場合には、CSSを使って行う必要があります(「font-variant」に「small-caps」を指定する)。

COLUMN

全角数字を半角数字に変換する

全角数字を半角数字に変換するには、次のように正規表現の「replace()」メソッドを使います。「replace()」メソッドの2番目のパラメータに関数を指定し、その関数内で処理することで全角数字を半角数字に変換することができます。

```
// 文字列を変数に代入
var srcString = "４５０１２３４５５333５５５６７８９を半角数字にする";
// 正規表現のreplace()を利用
var myArray = srcString.replace(/[０-９]/g, function(str, p1, p2, offset, s){
  return "０１２３４５６７８９".indexOf(str);
});
for(var myString="", i=0; i<myArray.length; i++){
  myString = myString + myArray[i];
}
```

関連項目 ▶▶▶

● 文字の形状を変更する ………………………… p.895
● 英文字の表示形式を指定する ………………… p.907

文字列の前後にある空白を削除する

ここでは、文字列の前後にある空白を削除する方法について解説します。

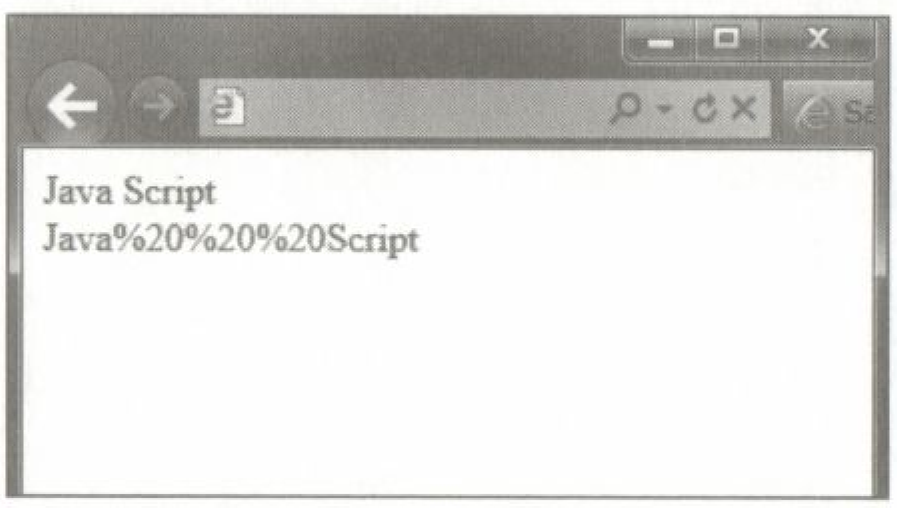
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入。末尾には全角空白が2つある
  var srcString = "　Java　　Script　　";
  // 前後の空白を削除
  var myString = srcString.trim();
  // 結果を出力
  ele.innerHTML = myString+"<br>"+escape(myString);
}, false);
```

ブラウザでindex.htmlを表示すると、前後の空白を削除した結果が表示されます。2行目は結果をわかりやすくするため、空白文字をエスケープしています。



ONEPOINT 文字列の前後の空白を削除するには「trim()」メソッドを使う

入力された文字の先頭にある空白や、文字列末尾にある余計な空白を削除するには、「trim()」メソッドを使います。実際には半角や全角空白だけでなく、タブ、改行コード、制御コードも削除対象となります。

SECTION-086

特定の文字列を削除する

ここでは、文字列の中から特定の文字列を削除する方法について解説します。

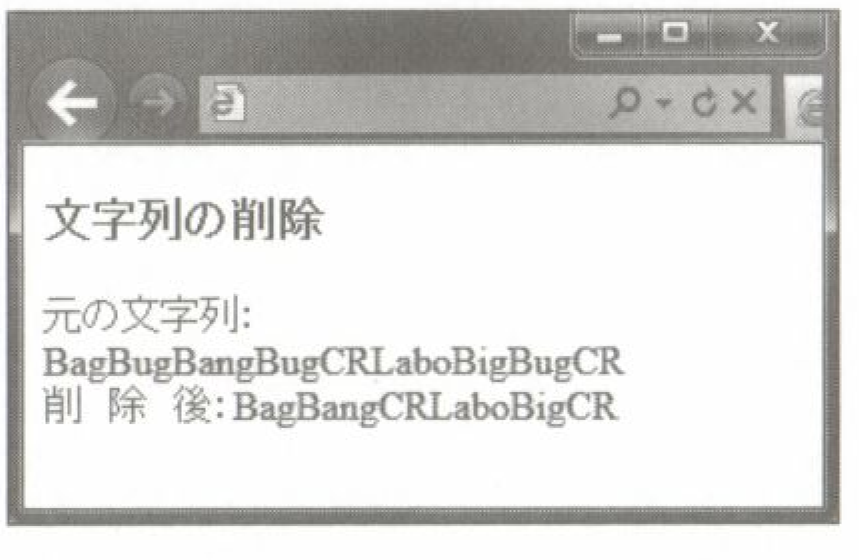
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>文字列の削除</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入。Bugの文字が3ヶ所にある
  var srcString = "BagBugBangBugCRLaboBigBugCR";
  // Bugの文字を削除した後に再度連結
  var myString = srcString.split("Bug").join("");
  // 結果を出力
  ele.innerHTML = "元の文字列 :"+srcString+"<br>削　除　後 :"+myString;
}, false);
```

ブラウザでindex.htmlを表示すると、文字列から「Bug」の文字列を削除した結果が表示されます。



ONEPOINT

文字列を削除するには「split()」メソッドのパラメータに削除したい文字を指定する

JavaScriptでは文字列を連結するには、「+」記号を使いますが、文字列を引くメソッドはありません。JavaScriptで文字列の中にある特定の文字列を削除するには、「split()」メソッドを使います。「split()」メソッドのパラメータに削除したい文字列を指定します。これにより、指定した文字で分割された結果が配列に入ります。この段階で指定した文字がなくなっていることになるので、後は「join()」メソッドを使って配列内容を連結します。これで指定した文字列が削除された結果を得ることができます。

COLUMN

文字列の数をカウントする

「split()」メソッドを使うと文字列の削除(文字列の引き算)ができますが、正規表現の「match()」メソッドを使うと削除すべき該当文字列の個数を求めることができます。次のようにすると、削除すべき「Bug」の文字の個数を表示します。

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 文字列を変数に代入。Bugの文字が3ヶ所にある
  var srcString = "BagBugBangBugCRLaboBigBugCR";
  // Bugの文字をカウント
  var n = srcString.match(/Bug/g).length;
  // 結果を出力
  ele.innerHTML = "削除すべきBugの数 :"+n;
}, false);
```

関連項目 ▶▶▶

- 文字列を指定文字で分割する……p.204
- 正規表現を使って文字を検索する……p.216

「RegExp」オブジェクトを生成する

ここでは、正規表現を扱う「RegExp」オブジェクトを生成する方法について解説します。

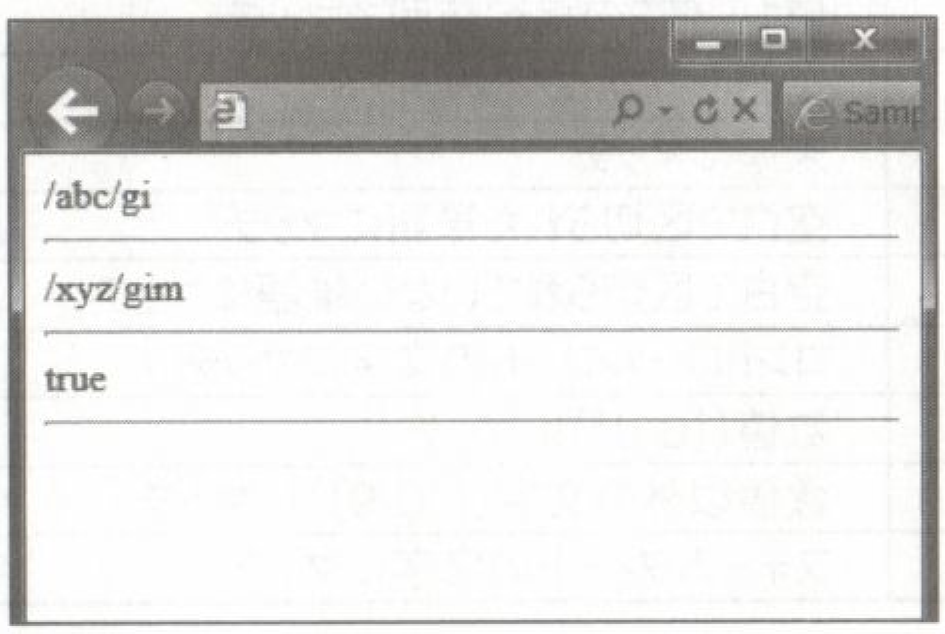
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output><hr>
    <output></output><hr>
    <output></output><hr>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // 正規表現オブジェクトを作成、その1
  var re1 = new RegExp("abc", "gi");
  // 正規表現オブジェクトを表示
  ele[0].innerHTML = re1;
  // 正規表現オブジェクトを作成、その2
  var re2 = /xyz/gim;
  // 正規表現オブジェクトを表示
  ele[1].innerHTML = re2;
  // 正規表現オブジェクトかどうか調べる
  ele[2].innerHTML = re2 instanceof RegExp;
}, false);
```

ブラウザでindex.htmlを表示すると、生成した正規表現オブジェクトの内容が表示されます。また、正規表現オブジェクトかどうかを調べた結果も表示されます。



ONEPOINT 「RegExp」オブジェクトを生成するには「new RegExp()」とする

JavaScriptでは、正規表現を使って複雑な文字列検索を行うことができます。正規表現オブジェクトを生成するには「new RegExp()」とし、パラメータには正規表現の文字列、2番目のパラメータにはオプションを指定します。指定できるオプションは、次の表のようになります。

オプション	説明
g	文字列全体にわたって検索
i	英文字の大文字小文字を無視
m	複数行を考慮

また、「new RegExp()」を使わずに、次のように「/」と「/」で囲んでも正規表現オブジェクトを生成することができます。

```
/abc/
/\d\d/
```

この場合、オプションは最後の「/」の後ろに指定します。

```
/abc/g
/\d\d/gim
```

COLUMN 正規表現文字の一覧

正規表現文字の一覧は、次の表のようになります。

正規表現文字	説明
文字	指定文字にマッチ
.	任意文字にマッチ(改行は除く)
[文字]	[]内にある任意文字にマッチ
[^文字]	[]内以外の任意文字にマッチ
文字|文字	|で区切られた任意文字にマッチ
文字*	0個以上の出現でマッチ
文字+	1個以上の出現でマッチ
文字?	0または1個の出現でマッチ
文字{値}	値の指定個数でマッチ
文字{値,}	値の指定個数以上でマッチ
文字{値1,値2}	値1～値2の指定範囲でマッチ
^文字	先頭の文字にマッチ
文字$	末尾にマッチ
文字\b	空白で区切られた単語にマッチ
文字\B	空白で区切られていない単語にマッチ
\c文字	コントロールコードの文字にマッチ
\d	数値([0-9])にマッチ
\D	数値以外の文字([^0-9])にマッチ
\f	フォームフィードの文字にマッチ

正規表現文字	説明
\n	改行文字にマッチ
\r	復帰文字にマッチ
\s	1文字の区切り文字([\f\n\r\t\v])にマッチ
\S	区切り文字以外([\f\n\r\t\v])の1文字にマッチ
\t	タブ文字にマッチ
\v	垂直タブ文字にマッチ
\w	英数文字([A-Za-z0-9_])にマッチ
\W	英数文字([A-Za-z0-9_])以外の文字にマッチ
\★	★(数値)番目の文字列(...)にマッチ
\o値	8進数の値にマッチ
\x値	16進数の値にマッチ
\.	.の文字
\-	-の文字
\\	\の文字
\/	/の文字
()	グループ化(マッチデータはRegExp.$1～9で取り出せる)

また、パターンマッチのサンプルは、次のようになります。

用例	説明
^\d+$	すべて数値の場合にマッチ(0123456789のみで構成されている)
^[a-z|A-Z]+$	アルファベットにマッチ(大文字小文字は正規表現のiオプションでも指定可能)
\d\d/\d\d	mm/ddにマッチ(ただし、123/456にもマッチするので注意)
\d{4}/\d{2}/\d{2}	yyyy/mm/ddにマッチ(ただし、1237459/65/12349にもマッチするので注意)
^\d{4}/\d{2}/\d{2}$	文字がyyyy/mm/ddのみの場合だけにマッチ(12345/65/123にはマッチしない)
^([01]?[0-9]|2[0-3]):([0-5][0-9])$	時間のHH:MMにマッチ(32:40や123:34、0:123にはマッチしない)
^〒\d{3}-\d{4}$	郵便番号にマッチ(〒XXX-XXXXのように3桁とハイフン区切りの4桁の数値で構成されている)
[ぁ-ん|ァ-ン]	平仮名か片仮名にマッチ
[\u3400-\u4DBF\u4E00-\u9FFF\uD840-\uD87F\uDC00-\uDFFF\uF900-\uFAFF]+	漢字にマッチ

関連項目 ▶▶▶

●正規表現を使って文字を検索する …… p.216
●正規表現を使って文字を置換する …… p.218

正規表現を使って文字を検索する

ここでは、正規表現を使って文字を検索する方法について解説します。

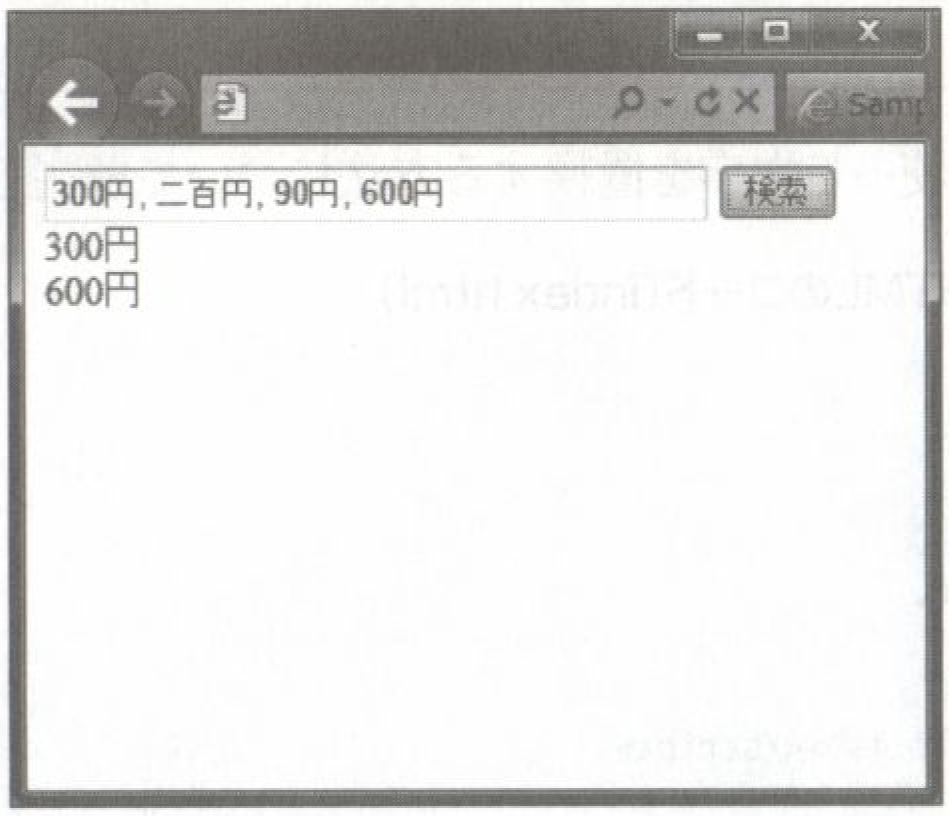
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="text" value="300円, 二百円, 90円, 600円" size="40" id="srcText">
      <input type="button" value="検索" id="findBtn">
    </form>
    <output></output>
  </body>
</html>
```

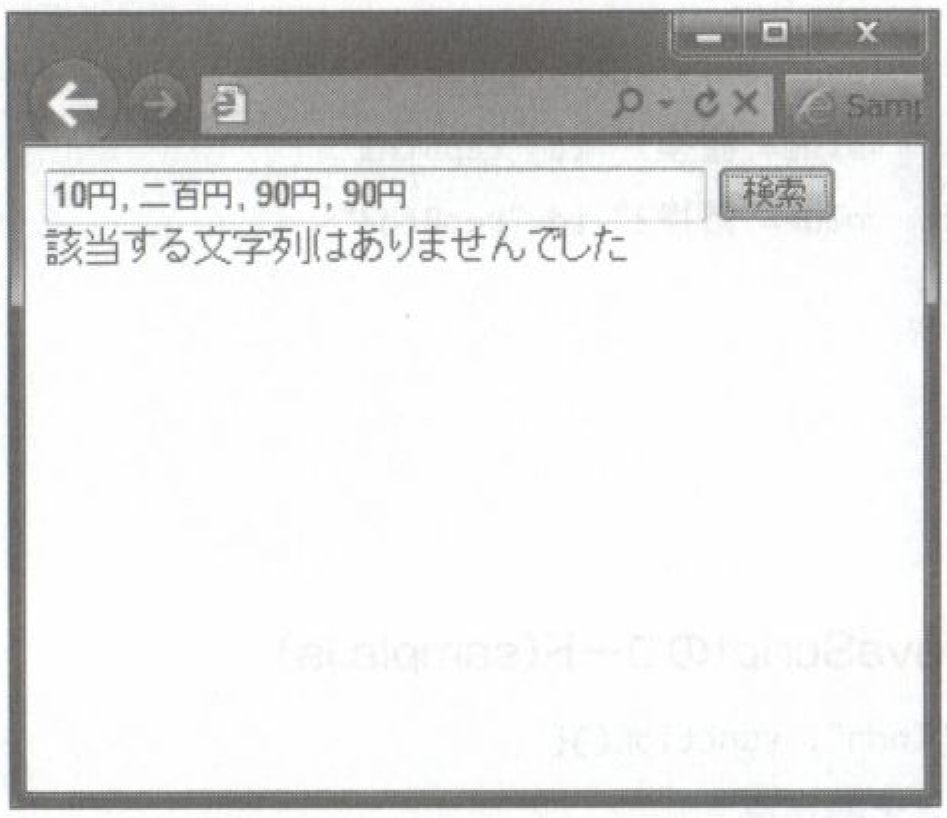
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを割り当て
  document.getElementById("findBtn").addEventListener("click", function(){
    // 検索対象文字列を読み出し
    var text = document.getElementById("srcText").value;
    // 正規表現で検索
    var list = text.match(/\d{3}円/g);
    // 結果を表示
    var ele = document.getElementsByTagName("output")[0];
    if (list){
      // マッチする文字列があった場合
      for(var i=0; i<list.length; i++){
        ele.innerHTML += list[i]+"<br>";
      }
    }else{
      // マッチする文字列がなかった場合
      ele.innerHTML = "該当する文字列はありませんでした";
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに検索対象文字列を指定して「検索」ボタンをクリックすると、マッチした結果が表示されます。



マッチする文字列がなかった場合は、異なるメッセージが表示されます。



ONEPOINT 正規表現を使って検索するには「match()」メソッドを使う

正規表現を使って検索するには、「match()」メソッドを使います。「match()」メソッドのパラメータには、正規表現オブジェクトを指定します。マッチする文字列があった場合、結果は配列で返されます。もし、マッチする文字列がなかった場合には、「match()」は「null」を返します。

関連項目 ▶▶▶

- 配列要素を検索する ………… p.134
- 文字列を検索する ………… p.200
- 「RegExp」オブジェクトを生成する ………… p.213
- 正規表現を使って文字を置換する ………… p.218

SECTION-089

正規表現を使って文字を置換する

ここでは、正規表現を使って文字を置換する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="text" value="300円, 二百円, 90円, 600円" size="40" id="srcText">
      <input type="button" value="置換1" id="repBtn1">
      <input type="button" value="置換2" id="repBtn2">
    </form>
    <output></output>
  </body>
</html>
```

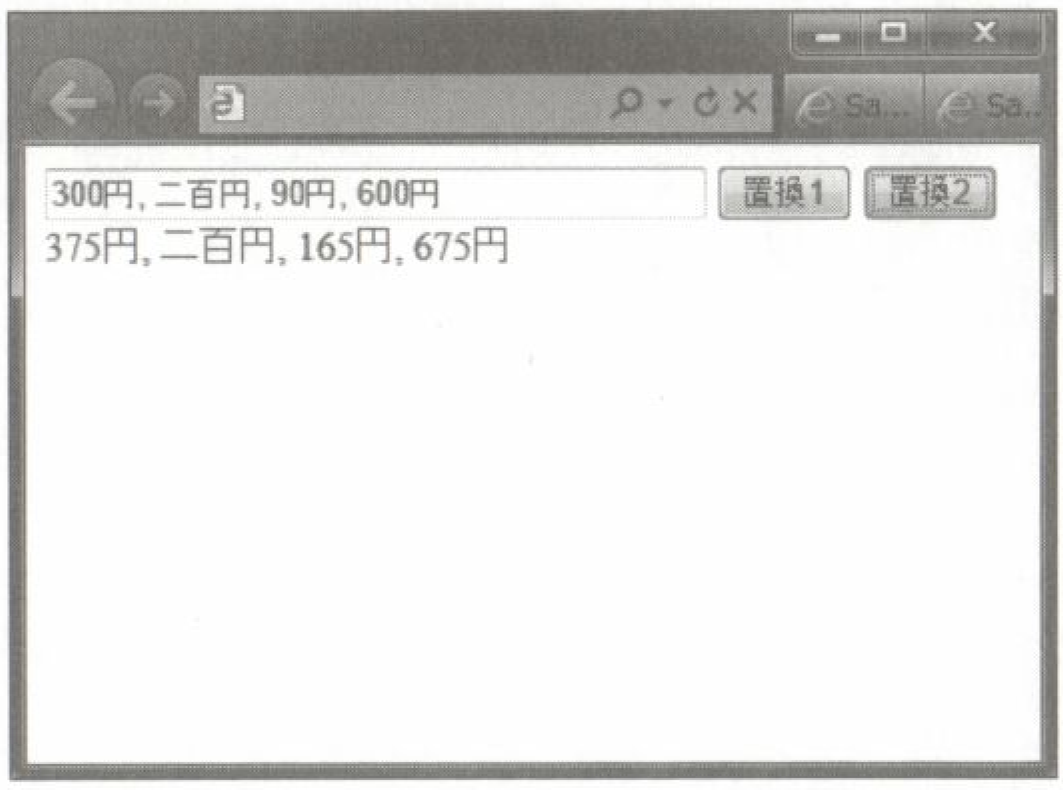
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 置換1ボタンにイベントを割り当て
  document.getElementById("repBtn1").addEventListener("click", function(){
    // 置換対象文字列を読み出し
    var text = document.getElementById("srcText").value;
    // 正規表現で置換
    var text = text.replace(/円/g, "万円");
    // 結果を表示
    document.getElementsByTagName("output")[0].innerHTML = text;
  }, false);
  // 置換2ボタンにイベントを割り当て
  document.getElementById("repBtn2").addEventListener("click", function(){
    // 置換対象文字列を読み出し
    var text = document.getElementById("srcText").value;
    // 正規表現で置換
    var text = text.replace(/\d{1,3}円/g, function(str, ptr, allStr){
      return parseInt(str) + 75 + "円";
    });
    // 結果を表示
    document.getElementsByTagName("output")[0].innerHTML = text;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに置換対象文字列を指定して、「置換1」ボタンをクリックすると、「円」が「万円」に置換されます。



「置換2」ボタンをクリックすると、マッチした数値に「75」が加算された結果が表示されます。



ONEPOINT 正規表現を使って文字を置換するには「replace()」メソッドを使う

正規表現を使って文字を置換するには、「replace()」メソッドを使います。「replace()」メソッドの最初のパラメータには、正規表現オブジェクトを指定します。2番目のパラメータには、置換する文字列を指定します。

IE9以降と他のブラウザ(Safari/Google Chrome/Firefox/Opera)では、「replace()」メソッドの2番目のパラメータに関数を指定することができます。関数からの戻り値が置換文字列として処理されます。この関数には「正規表現文字列のグループ指定である()の個数+3」個の数だけパラメータが渡されます。3つのパラメータは「マッチした文字」「マッチした位置」「検索対象文字」の順番になります。サンプルのように「()」によるグループ指定がない場合は、「マッチした文字」「マッチした位置」「検索対象文字」が渡されることになります。

関連項目 ▶▶▶

● 「RegExp」オブジェクトを生成する …… p.213
● 正規表現を使って文字を検索する …… p.216

CHAPTER 06

日付・時刻

「Date」オブジェクトを生成する

ここでは、「Date」オブジェクトを生成する方法について解説します。

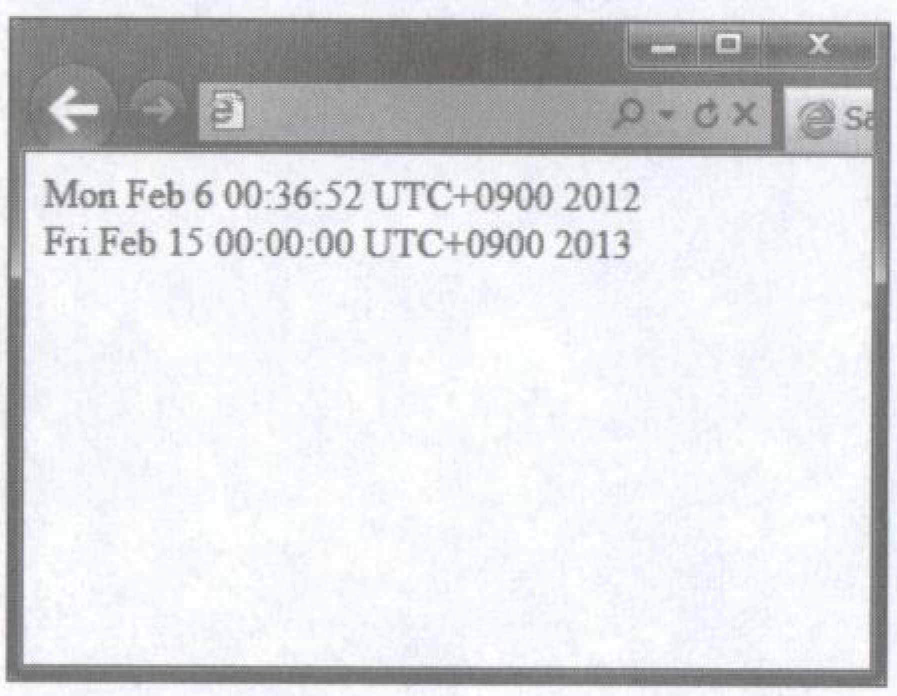
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var date1 = new Date();
  // 2013年2月15日を示すDateオブジェクトを生成
  var date2 = new Date("2013/2/15");
  // 結果を出力
  ele.innerHTML = date1+"<br>"+date2;
}, false);
```

ブラウザでindex.htmlを表示すると、「Date」オブジェクトの生成結果が表示されます。



ONEPOINT 日付を扱うには「Date」オブジェクトを使う

JavaScriptで日付を扱うには、「Date」オブジェクトを利用します。「Date」オブジェクトを生成する際にパラメータを指定しないと、生成時点での日時情報が格納されます。パラメータに日付や時間を示す文字列を指定すると、その日付情報を持つオブジェクトが生成されます。なお、「Date」オブジェクトを文字列に変換して表示した場合、ブラウザによって結果が異なります。

関連項目 ▶▶▶

- 日付を求める……p.224
- 時刻を求める……p.226
- 協定世界時の日付を求める……p.228
- 協定世界時の時刻を求める……p.230
- 日付を設定する……p.234
- 時刻を設定する……p.236
- 協定世界時の日付を設定する……p.238
- 協定世界時の時刻を設定する/時差を求める……p.240

日付を求める

ここでは、日付を求める方法について解説します。

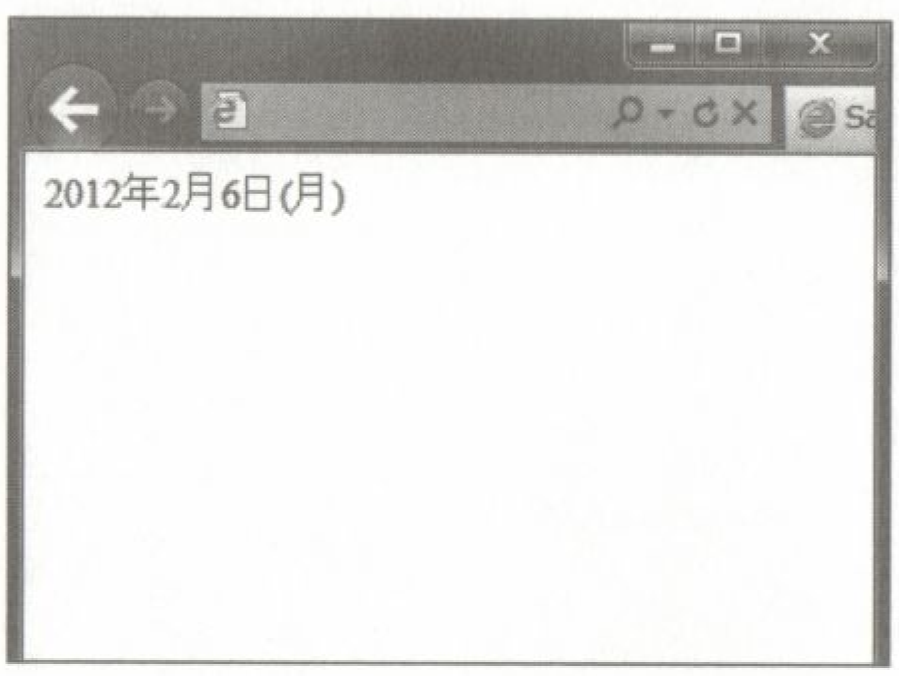
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 西暦年4桁を読み出し
  var y = currentDate.getFullYear();
  // 月を読み出し
  var m = currentDate.getMonth() + 1;
  // 日にちを読み出し
  var d = currentDate.getDate();
  // 曜日(0〜6)を読み出し
  var n = currentDate.getDay();
  var day = "日月火水木金土".charAt(n);
  // 結果を出力
  ele.innerHTML = y+"年"+m+"月"+d+"日("+day+")";
}, false);
```

ブラウザでindex.htmlを表示すると、現在の日付が表示されます。



ONEPOINT

日付を読み出すには「getYear()」メソッドや「getDate()」メソッドなどを使う

JavaScriptで日付情報を読み出すには、「Date」オブジェクトを生成します。生成時点での日付を取得したい場合はパラメータには何も指定せず、「Date」オブジェクトを生成します。特定の日付の情報を得たい場合には、「Date」オブジェクト生成時に日付を示す文字列を指定するか、「setFullYear()」などのメソッドを使って設定します。生成した「Date」オブジェクトから次のメソッドを使うことで日付情報を読み出すことができます。なお、月を読み出す「getMonth()」メソッドは実際の月よりも「1」少ない値になります。

メソッド	説明
getYear()	西暦年を返す
getFullYear()	西暦年4桁を返す
getMonth()	月を返す(実際の月よりも1少ない値を返す)
getDate()	日にちを返す
getDay()	曜日を返す。曜日は0～6までの数値で次のように対応している 0:日曜日 1:月曜日 2:火曜日 3:水曜日 4:木曜日 5:金曜日 6:土曜日

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する …… p.222
- 時刻を求める …… p.226
- 協定世界時の日付を求める …… p.228
- 協定世界時の時刻を求める …… p.230

時刻を求める

ここでは、時刻を求める方法について解説します。

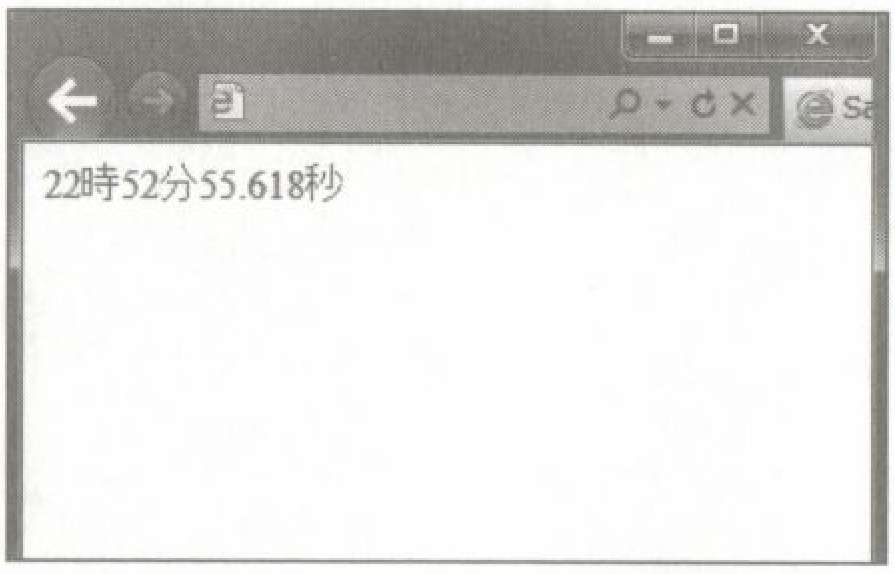
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  setInterval(function(){
    var ele = document.getElementsByTagName("output")[0];
    // 現在の日時を示すDateオブジェクトを生成
    var currentDate = new Date();
    // 時間(0～23)を読み出し
    var h = currentDate.getHours();
    // 分(0～59)を読み出し
    var m = currentDate.getMinutes();
    // 秒(0～59)を読み出し
    var s = currentDate.getSeconds();
    // ミリ秒(0～999)を読み出し
    var ms = currentDate.getMilliseconds();
    // 結果を出力
    ele.innerHTML = h+"時"+m+"分"+s+"."+ms+"秒";
  }, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、現在の時刻がリアルタイムに表示されます。



ONEPOINT

時刻を読み出すには「getHours()」メソッドや「getMinutes()」メソッドを使う

JavaScriptで時刻を読み出すには「Date」オブジェクトを生成します。生成時点での時刻を取得したい場合はパラメータには何も指定せず、「Date」オブジェクトを生成します。時計などのリアルタイムに時刻を表示するような場合は、その都度、「new Date()」としてオブジェクトを生成し、そのオブジェクトから時刻を読み出す必要があります。

「Date」オブジェクトの時刻に関するメソッドは、次の表のようになります。

メソッド	説明
getHours()	時を返す。24時間制なので0～23までの整数値になる
getMinutes()	分を返す。0～59までの整数値になる
getSeconds()	秒を返す。0～59までの整数値になる
getMilliseconds()	ミリ秒を返す。0～999までの整数値になる

COLUMN

JavaScriptの「Date」オブジェクトで扱える日付の限界

JavaScriptの「Date」オブジェクトは、1970年1月1日午前0時からの秒数を元にして処理が行われます。これはJavaScript独自の処理ではなく、UNIXで使用されている日付/時刻処理を利用しているためです。

この時刻処理は32ビット長で秒数をカウントするため、協定世界時の2038年1月19日3時14分7秒を越えると正しい日付処理が行えない問題があります。

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する……p.222
- 日付を求める……p.224
- 協定世界時の日付を求める……p.228
- 協定世界時の時刻を求める……p.230

協定世界時の日付を求める

ここでは、協定世界時の日付を求める方法について解説します。

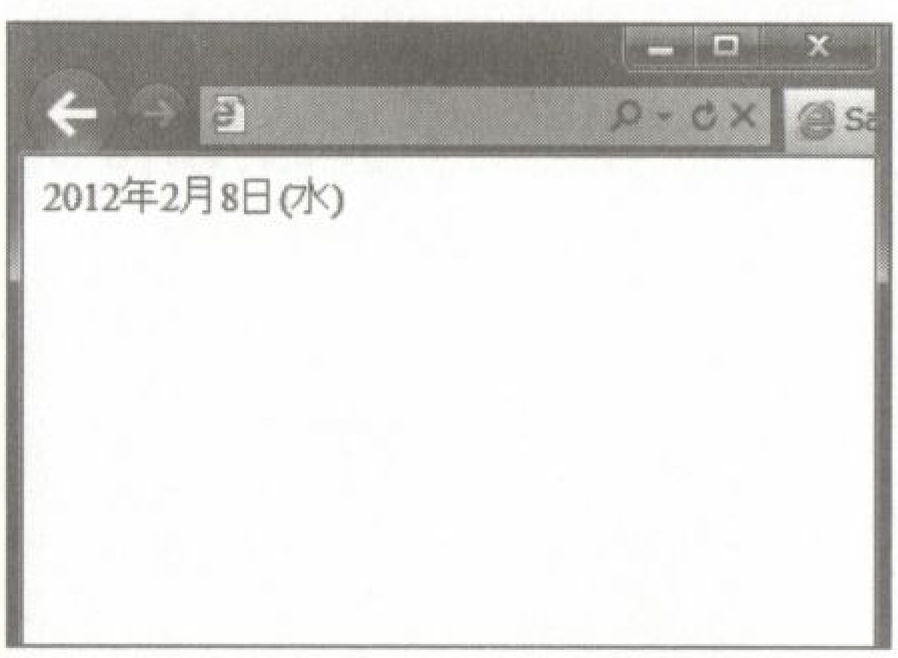
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 協定世界時の西暦年4桁を読み出し
  var y = currentDate.getUTCFullYear();
  // 協定世界時の月を読み出し
  var m = currentDate.getUTCMonth() + 1;
  // 協定世界時の日にちを読み出し
  var d = currentDate.getUTCDate();
  // 協定世界時の曜日(0～6)を読み出し
  var n = currentDate.getUTCDay();
  var day = "日月火水木金土".charAt(n);
  // 結果を出力
  ele.innerHTML = y+"年"+m+"月"+d+"日("+day+")";
}, false);
```

ブラウザでindex.htmlを表示すると、現在の協定世界時の日付が表示されます。



ONEPOINT 協定世界時を読み出すには「getUTCFullYear()」メソッドや「getUTCDate()」メソッドなどを使う

JavaScriptで協定世界時の日付情報を読み出すには、「Date」オブジェクトを生成します。生成時点での協定世界時の日付を取得したい場合はパラメータには何も指定せず、「Date」オブジェクトを生成します。特定の協定世界時の日付の情報を得たい場合には、「Date」オブジェクト生成時に日付を示す文字列を指定するか、「setUTCFullYear()」などのメソッドを使って設定します。生成した「Date」オブジェクトから次のメソッドを使うことで、協定世界時の日付情報を読み出すことができます。なお、協定世界時の月を読み出す「getUTCMonth()」メソッドは、実際の月よりも1少ない値になります。

メソッド	説明
getUTCFullYear()	協定世界時の西暦年4桁を返す
getUTCMonth()	協定世界時の月を返す（実際の月よりも1少ない値を返す）
getUTCDate()	協定世界時の日にちを返す
getUTCDay()	協定世界時の曜日を返す。曜日は0～6までの数値で次のように対応している。 0:日曜日 1:月曜日 2:火曜日 3:水曜日 4:木曜日 5:金曜日 6:土曜日

COLUMN グリニッジ標準時

協定世界時はグリニッジ標準時と同じです。協定世界時ではUTC、グリニッジ標準時ではGMTと略されます。JavaScriptのメソッドでもGMTの名残りがあります。

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する ………… p.222
- 日付を求める ………… p.224
- 時刻を求める ………… p.226
- 協定世界時の時刻を求める ………… p.230
- グリニッジ標準時を求める ………… p.244

SECTION-094

協定世界時の時刻を求める

ここでは、協定世界時の時刻を求める方法について解説します。

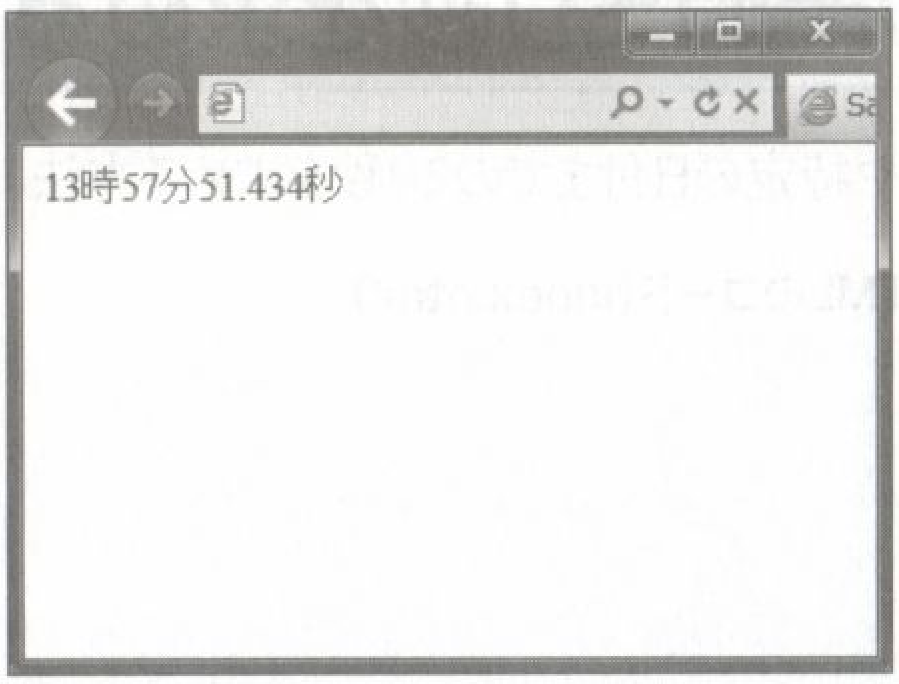
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  setInterval(function(){
    var ele = document.getElementsByTagName("output")[0];
    // 現在の日時を示すDateオブジェクトを生成
    var currentDate = new Date();
    // 協定世界時の時間(0～23)を読み出し
    var h = currentDate.getUTCHours();
    // 協定世界時の分(0～59)を読み出し
    var m = currentDate.getUTCMinutes();
    // 協定世界時の秒(0～59)を読み出し
    var s = currentDate.getUTCSeconds();
    // 協定世界時のミリ秒(0～999)を読み出し
    var ms = currentDate.getUTCMilliseconds();
    // 結果を出力
    ele.innerHTML = h+"時"+m+"分"+s+"."+ms+"秒";
  }, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、現在の協定世界時の時刻がリアルタイムに表示されます。



ONEPOINT 協定世界時の時刻を読み出すには「getUTCHours()」メソッドや「getUTCMinutes()」メソッドなどを使う

JavaScriptで協定世界時の時刻を読み出すには、「Date」オブジェクトを生成します。生成時点での協定世界時の時刻を取得したい場合はパラメータには何も指定せず、「Date」オブジェクトを生成します。「Date」オブジェクトの協定世界時の時刻に関するメソッドを次の表に示します。日本の場合、協定世界時よりも9時間進んだ時刻になります(日本から見れば協定世界時は9時間前の時刻になる)。

メソッド	説明
getUTCHours()	協定世界時の時を返す。24時間制なので0～23までの整数値になる
getUTCMinutes()	協定世界時の分を返す。0～59までの整数値になる
getUTCSeconds()	協定世界時の秒を返す。0～59までの整数値になる
getUTCMilliseconds()	協定世界時のミリ秒を返す。0～999までの整数値になる

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する……p.222
- 日付を求める……p.224
- 時刻を求める……p.226
- 協定世界時の日付を求める……p.228
- グリニッジ標準時を求める……p.244

IE 6 Chrome Safari 5 Firefox 4 Opera 11 iOS 4 Android 2 WP

SECTION-095

1970年1月1日午前0時0分0秒から指定時までの経過ミリ秒を求める

ここでは、現在のミリ秒や特定の日付までのミリ秒を求める方法について解説します。

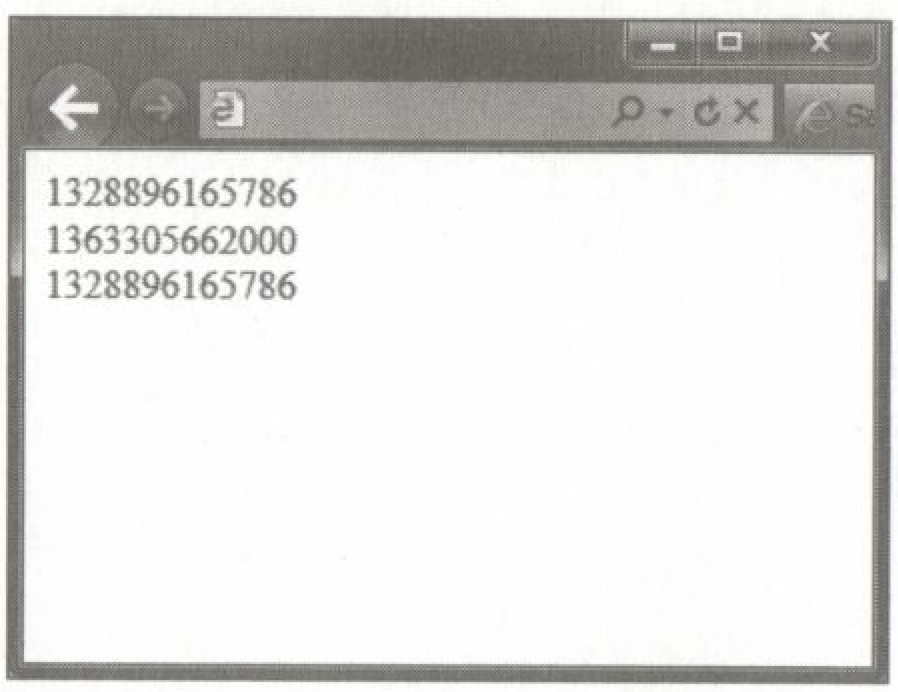
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 1970年1月1日から現時点までの経過ミリ秒を読み出し
  var msec1 = currentDate.getTime();
  // 1970年1月1日から2013年2月15日0時1分2秒までの経過ミリ秒を読み出し
  var msec2 = Date.UTC(2013, 2, 15, 0, 1, 2);
  // 1970年1月1日から現時点までの経過ミリ秒を読み出し
  var msec3 = Date.now();
  // 結果を出力
  ele.innerHTML = msec1+"<br>"+msec2+"<br>"+msec3;
}, false);
```

ブラウザでindex.htmlを表示すると、指定時までのミリ秒が表示されます。



ONEPOINT 経過ミリ秒を取得するには「Date.now()」や「getTime()」メソッドを使う

JavaScriptでは、1970年1月1日午前0時0分0秒から指定時までの経過ミリ秒を求めることができます。現時点までの経過ミリ秒は、「Date.now()」として求めることができます。ただし、「now()」メソッドは、IE6などの古いブラウザではサポートされていません。「now()」メソッドと同様に、現時点までの経過ミリ秒を求めるには、「Date」オブジェクトの「getTime()」メソッドを使います。「Date.now()」と比べて、「Date」オブジェクトを生成する手間がかかります。

また、日付や時間を指定して経過ミリ秒を求めたい場合には、「Date.UTC()」メソッドを使います。パラメータには年月日、時分秒を指定します。

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する ………… p.222

SECTION-096

日付を設定する

ここでは、日付を設定する方法について解説します。

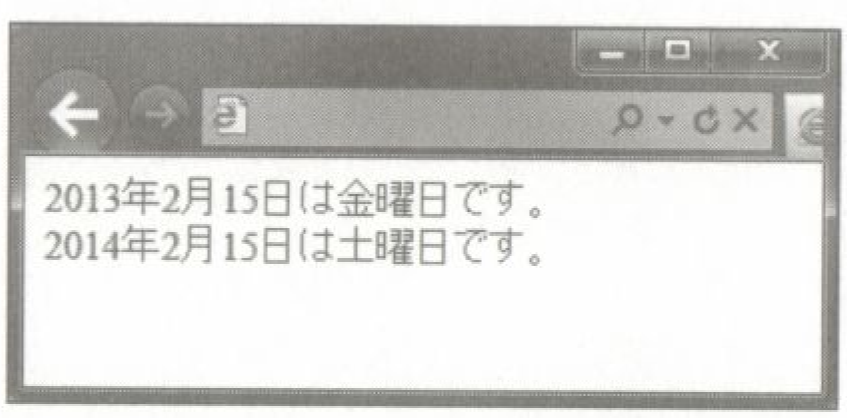
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 2013年2月15日に設定
  currentDate.setFullYear(2013);
  currentDate.setMonth(1);
  currentDate.setDate(15);
  // 曜日を読み出し
  var dw1 = "日月火水木金土".charAt(currentDate.getDay());
  // 日付を指定してDateオブジェクトを生成
  var myDate = new Date("2014/2/15");
  // 曜日を読み出し
  var dw2 = "日月火水木金土".charAt(myDate.getDay());
  // 結果を出力
  ele.innerHTML = "2013年2月15日は"+dw1+"曜日です。<br>";
  ele.innerHTML += "2014年2月15日は"+dw2+"曜日です。";
}, false);
```

ブラウザでindex.htmlを表示すると、設定した日付の曜日が表示されます。



ONEPOINT

日付を設定するには「setFullYear()」「setMonth()」「setDate()」メソッドを使う

JavaScriptでは、生成した「Date」オブジェクトに用意されたメソッドを使うことで日付を生成することができます。日付を設定できるメソッドは、次の表のようになります。

メソッド	説明
setFullYear()	西暦年4桁
setMonth()	月。「0」が1月を示し、「11」であれば12月を示す
setDate()	日にち

パラメータは、範囲を超えた場合でもエラーにはなりません。たとえば、「setMonth(12)」とすると、次の年の1月に設定されます。日にちも同様で「setDate(60)」とすると、60日先の日付が設定されます(負数も使用できる)。日付の設定メソッドは、生成した日時によっては期待通り設定されないことがあります。このような場合は、サンプルのように「Date」オブジェクトを生成する際に日付文字列を設定します。

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する ……… p.222
● 時刻を設定する ……… p.236
● 協定世界時の日付を設定する ……… p.238
● 協定世界時の時刻を設定する/時差を求める ……… p.240

時刻を設定する

ここでは、時刻を設定する方法について解説します。

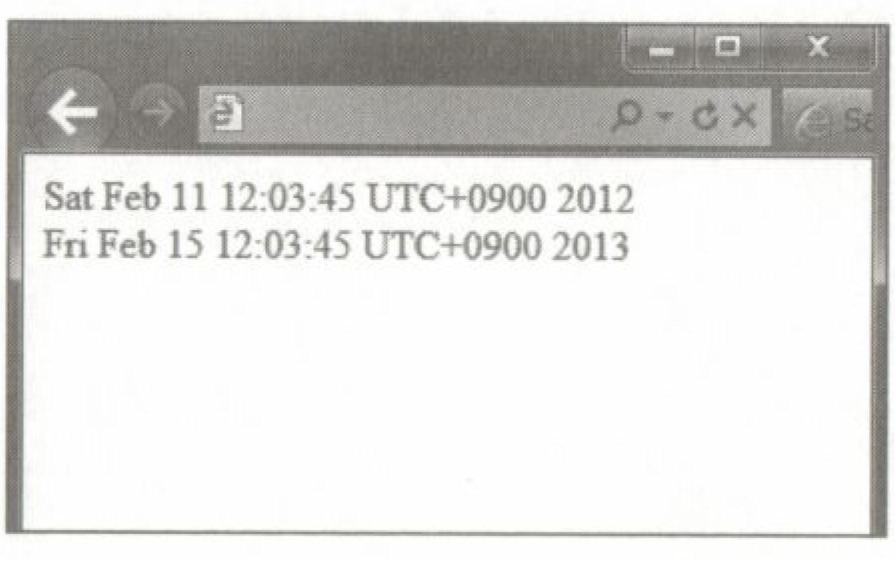
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 12時03分45.678秒設定
  currentDate.setHours(12);
  currentDate.setMinutes(3);
  currentDate.setSeconds(45);
  currentDate.setMilliseconds(678);
  // 時間を指定してDateオブジェクトを生成
  var myDate = new Date("Sat Feb 15 2013 12:03:45 GMT+0900 (JST)");
  // 結果を出力
  ele.innerHTML = currentDate+"<br>";
  ele.innerHTML += myDate;
}, false);
```

ブラウザでindex.htmlを表示すると、設定した時刻の「Date」オブジェクト情報が表示されます。なお、「Date」オブジェクトを文字列に変換して表示した場合、ブラウザによって結果が異なります。



ONEPOINT

時刻を設定するには「setHours()」メソッドや「setMinutes()」メソッドなどを使う

JavaScriptでは、生成した「Date」オブジェクトに用意されたメソッドを使うことで時刻を生成することができます。時刻を設定できるメソッドには、次の表のようになります。

メソッド	説明
setHours()	時
setMinutes()	分
setSeconds()	秒
setMilliseconds()	ミリ秒

パラメータは、範囲を超えた場合でもエラーにはなりません。たとえば、「setMinutes(90)」とすると、「Date」オブジェクトで示される時刻に90分が加算されます。これは他のメソッドでも同様です。正数だけでなく負数も指定できます。「setSeconds(-100)」だと100秒前に設定されます。

また、日付と時刻を同時に設定したい場合は、サンプルのように、「Date」オブジェクトを生成する際に日付と時刻を示す文字列を設定します。

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する …… p.222
- 日付を設定する …… p.234
- 協定世界時の日付を設定する …… p.238
- 協定世界時の時刻を設定する/時差を求める …… p.240

協定世界時の日付を設定する

ここでは、協定世界時の日付を設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 2013年2月15日に設定
  currentDate.setUTCFullYear(2013);
  currentDate.setUTCMonth(1);
  currentDate.setUTCDate(15);
  // 曜日を読み出し
  var dw = "日月火水木金土".charAt(currentDate.getUTCDay());
  // 結果を出力
  ele.innerHTML = "2013年2月15日は"+dw+"曜日です。";
}, false);
```

ブラウザでindex.htmlを表示すると、設定した協定世界時の日付の曜日が表示されます。



ONEPOINT

協定世界時の日付を設定するには「setUTCFullYear()」「setUTCMonth()」「setUTCDate()」メソッドを使う

JavaScriptでは、生成した「Date」オブジェクトに用意されたメソッドを使うことで協定世界時の日付を生成することができます。協定世界時の日付を設定できるメソッドは、次の表のようになります。

メソッド	説明
setUTCFullYear()	西暦年4桁
setUTCMonth()	月。「0」が1月を示し、「11」であれば12月を示す
setUTCDate()	日にち

パラメータは、範囲を超えた場合でもエラーにはなりません。たとえば、「setUTCDate(12)」とすると、次の年の1月が設定されます。日にちも同様で、「setUTCDate(60)」とすると、60日先の日付が設定されます(負数も使用できる)。

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する ……… p.222
● 日付を設定する ……… p.234
● 時刻を設定する ……… p.236
● 協定世界時の時刻を設定する/時差を求める ……… p.240

IE 6 Chrome Safari 5 Firefox 4 Opera 11 iOS 4 Android 2 WP

SECTION-099

協定世界時の時刻を設定する/時差を求める

ここでは、協定世界時の時間を設定する/時差を求める方法について解説します。

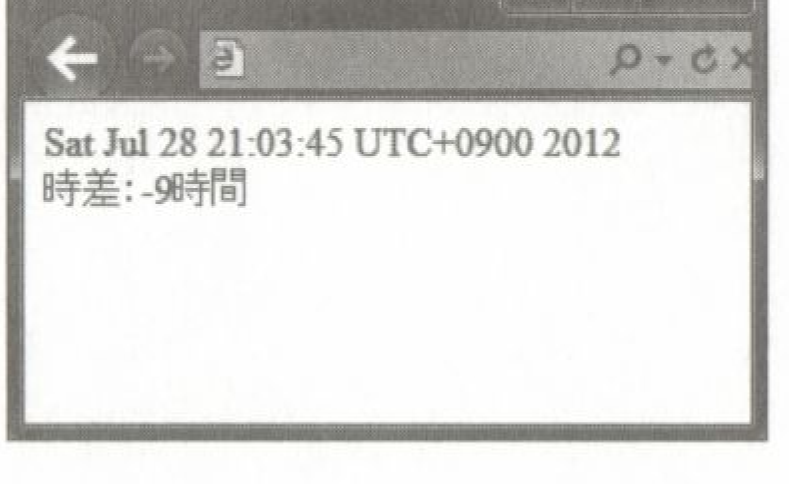
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 12時03分45.678秒設定
  currentDate.setUTCHours(12);
  currentDate.setUTCMinutes(3);
  currentDate.setUTCSeconds(45);
  currentDate.setUTCMilliseconds(678);
  // タイムゾーンを読み出し
  var timezone = currentDate.getTimezoneOffset();
  // 結果を出力
  ele.innerHTML = currentDate+"<br>";
  ele.innerHTML += "時差 :"+timezone/60+"時間";
}, false);
```

ブラウザでindex.htmlを表示すると、設定した協定世界時の時刻と時差が表示されます。なお、「Date」オブジェクトを文字列に変換して表示した場合、ブラウザによって結果が異なります。



ONEPOINT

協定世界時の時刻を設定するには「setUTCHours()」「setUTCMinutes()」メソッドなどを使う

JavaScriptでは、生成した「Date」オブジェクトに用意されたメソッドを使うことで協定世界時の時刻を生成することができます。協定世界時の時刻を設定できるメソッドには、次の表のようになります。

メソッド	説明
setUTCHours()	時
setUTCMinutes()	分
setUTCSeconds()	秒
setUTCMilliseconds()	ミリ秒

パラメータは、範囲を超えた場合でもエラーにはなりません。たとえば、「setUTCMinutes(90)」とすると、「Date」オブジェクトで示される時刻に90分が加算されます。これは他のメソッドでも同様です。正数だけでなく負数も指定できます。「setUTCSeconds(-100)」だと100秒前に設定されます。

日本と世界標準時には時差があります。この時差を求めるには、「getTimezoneOffset()」メソッドを使います。これは時差を分で返します。日本の場合は「-540」となります。「60」で除算すれば時差-9時間を求めることができます。

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する ……… p.222
● 日付を設定する ……… p.234
● 時刻を設定する ……… p.236
● 協定世界時の日付を設定する ……… p.238

さまざまな日付や時刻の形式に変換する

ここでは、さまざまな日付や時刻の形式に変換する方法について解説します。

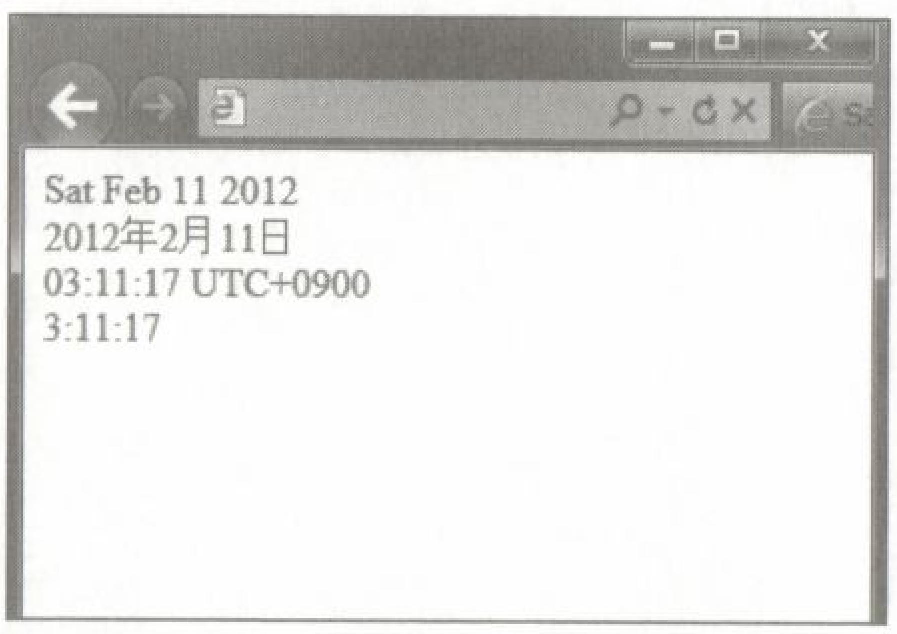
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // 日付を文字列に変換
  var dateStr = currentDate.toDateString();
  // 日付を地域文字列に変換
  var dateLStr = currentDate.toLocaleDateString();
  // 時刻を文字列に変換
  var timeStr = currentDate.toTimeString();
  // 時刻を地域文字列に変換
  var timeLStr = currentDate.toLocaleTimeString();
  // 結果を出力
  ele.innerHTML = dateStr+"<br>"+dateLStr+"<br>"+timeStr+"<br>"+timeLStr;
}, false);
```

ブラウザでindex.htmlを表示すると、日付と時刻が表示されます。また、地域に合わせた日付と時刻も表示されます。



ONEPOINT

地域に合わせた日付と時刻を取得するには「toLocateDateString()」メソッドや「toLocaleTimeString()」メソッドを使う

JavaScriptでは、共通の日付と時刻だけでなく、地域(ロケール)に合わせた日付や時刻も取得することができます。「Date」オブジェクトを日付文字に変換するには、「toDateString()」メソッドを使います。これによって日付だけが文字列化されます。時間の場合は、「toTimeString()」メソッドを使います。

地域に合わせた日付を取得したい場合には、「toLocateDateString()」メソッドや「toLocaleTimeString()」メソッドを使います。たとえば、「toLocaleTimeString()」メソッドでは、日本の場合は「2013年2月15日」といった文字列が返されます。

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する ………………………………………… p.222

SECTION-101

グリニッジ標準時を求める

ここでは、グリニッジ標準時を求める方法について解説します。

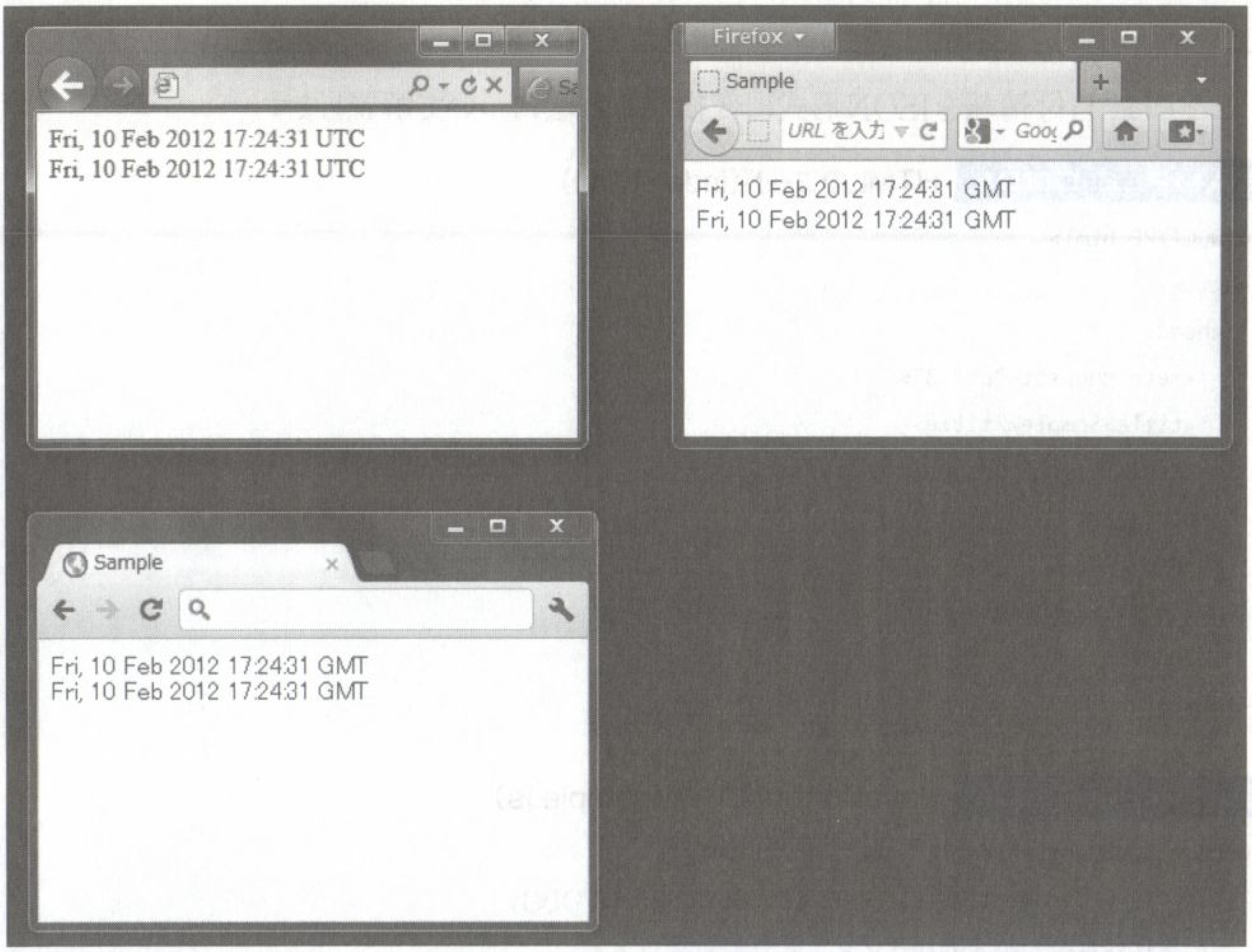
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  setTimeout(function(){
    var ele = document.getElementsByTagName("output")[0];
    // 現在の日時を示すDateオブジェクトを生成
    var currentDate = new Date();
    // グリニッジ標準時を読み出し
    var gmt = currentDate.toGMTString();
    // 協定世界時を読み出し
    var utc = currentDate.toUTCString();
    // 結果を出力
    ele.innerHTML = gmt+"<br>"+utc;
    setTimeout(arguments.callee, 1000);
  }, 1000);
}, false);
```

ブラウザでindex.htmlを表示すると、グリニッジ標準時と協定世界時の日時がリアルタイムに表示されます。



ONEPOINT **グリニッジ標準時を求めるには「toGMTString()」メソッドを使う**

JavaScriptでグリニッジ標準時(GMT:Greenwich Mean Time)を求めるには、「Date」オブジェクトの「toGMTString()」メソッドを利用します。グリニッジ標準時は、現在では協定世界時に置き換えられています。JavaScriptではグリニッジ標準時も協定世界時も同じ結果を返します。なお、協定世界時を求めるには、「Date」オブジェクトで「UTC」の文字がついたメソッドを利用します。文字列として取得したい場合には、「Date」オブジェクトの「toUTCString()」メソッドを使います。

COLUMN **GMTなのにUTCと表示される**

IEでは、「toGMTString()」を呼び出しても、返される文字列には「GMT」が含まれず、「UTC」になります(上記の結果画像の左上を参照)。

関連項目 ▶▶▶

- 「Date」オブジェクトを生成する ………… p.222
- 協定世界時の日付を求める ………… p.228
- 協定世界時の時刻を求める ………… p.230

SECTION-102

日付情報をJSON形式に変換する

ここでは、日付情報をJSON形式に変換する方法について解説します。

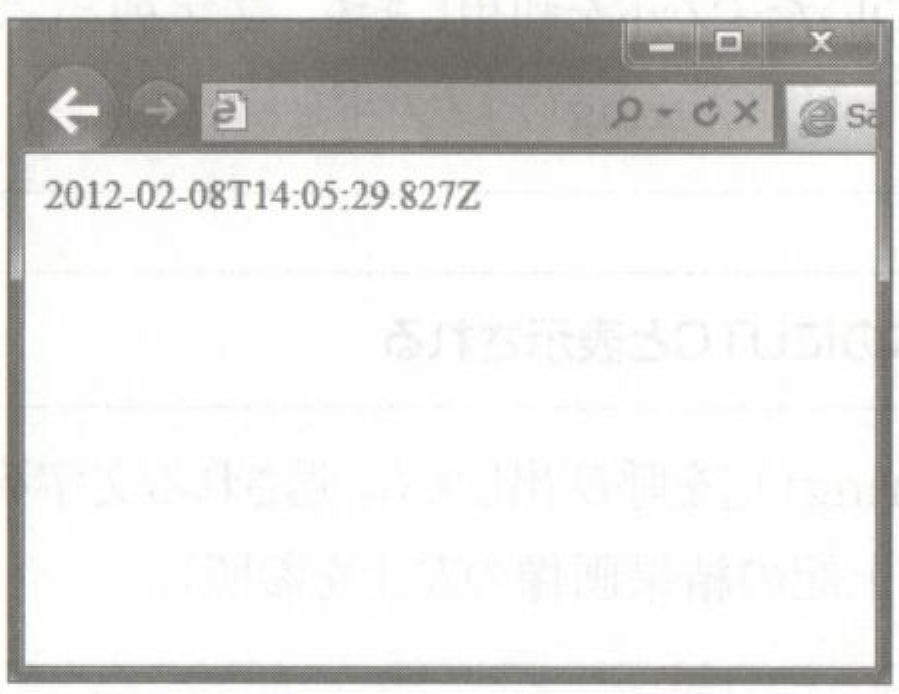
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 現在の日時を示すDateオブジェクトを生成
  var currentDate = new Date();
  // JSON形式に変更
  var jsonStr = currentDate.toJSON();
  // 結果を出力
  ele.innerHTML = jsonStr;
}, false);
```

ブラウザでindex.htmlを表示すると、JSON形式に変換した日付が文字列で表示されます。



ONEPOINT 日付をJSON形式に変換するには「toJSON()」メソッドを使う

日付情報をJSON形式に変換するには、「toJSON()」メソッドを使います。「toJSON()」メソッドを使うと、次の形式に変換されます。

```
yyyy-mm-ddTHH:MM:SS.NNNZ
```

それぞれ、次の日付情報を表します。

記号	説明
yyyy	西暦年4桁
mm	月
dd	日にち
HH	時(24時間制)
MM	分
SS	秒
NNN	ミリ秒
T	以後の文字が時刻を示す
Z	時刻データの末尾を示す

たとえば、「2012年2月8日14時7分3分314秒」であれば、「2012-02-08T14:07:03.314Z」となります。

COLUMN JSONについて

JSONはJavaScript Object Notationの略で、読みやすくコンパクトな形式です。XML形式よりも手軽に扱えるため、XMLに変わってさまざまな場面で利用されています。JSON形式については、次のページに仕様が掲載されています。

URL http://www.json.org/json-ja.html

関連項目 ▶▶▶

● 「Date」オブジェクトを生成する …… p.222

CHAPTER 07
ブラウザオブジェクト

SECTION-103

ブラウザ名やバージョンを取得する

ブラウザによって動作に違いがあるため、どの環境で動作しているかを調べる必要が出てくる場合があります。ここでは、ブラウザの名前やバージョンを取得する方法について解説します。

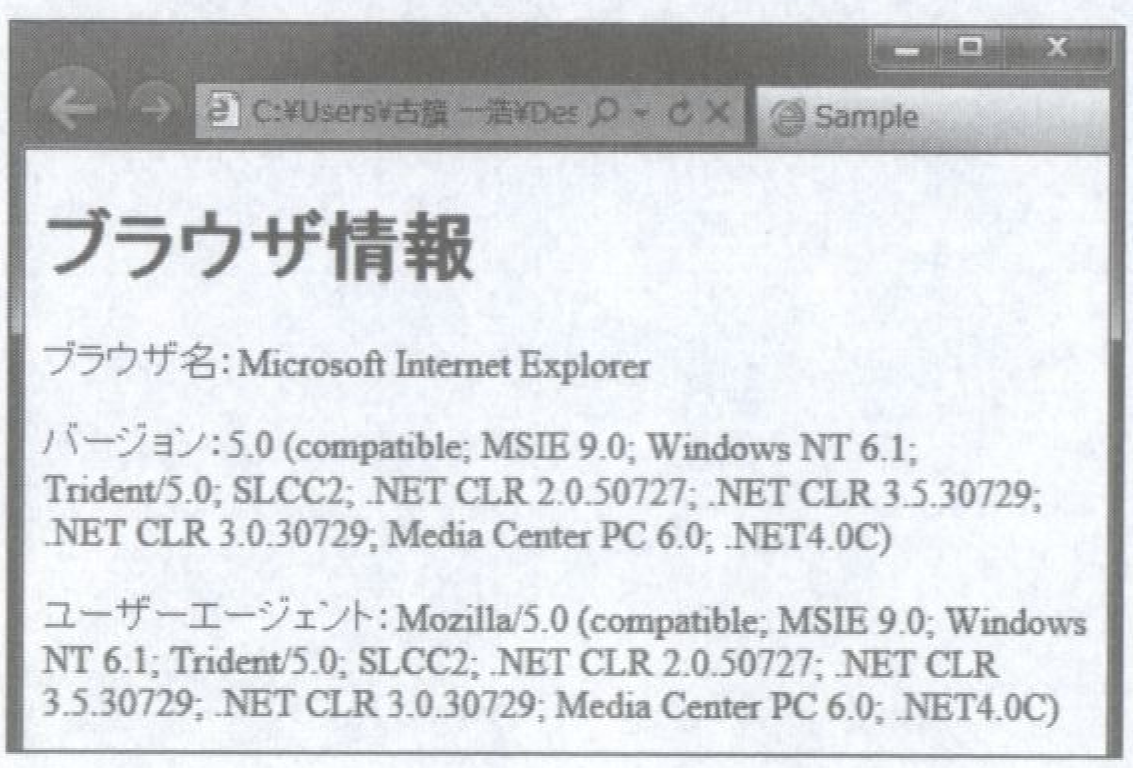
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ブラウザ情報</h1>
    <p>ブラウザ名 :<output></output></p>
    <p>バージョン :<output></output></p>
    <p>ユーザーエージェント :<output></output></p>
  </body>
</html>
```

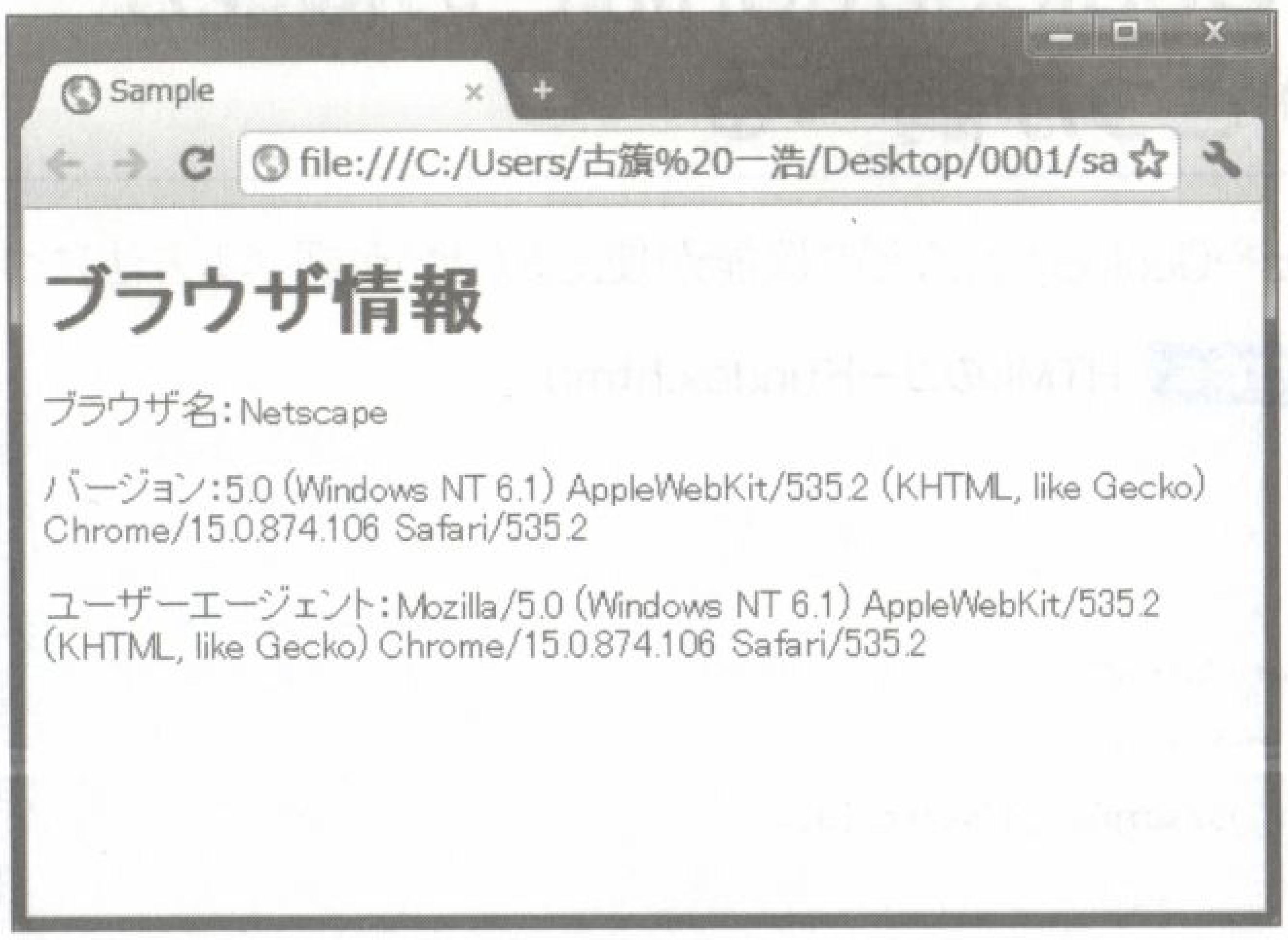
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // ブラウザ名を表示
  ele[0].innerHTML = navigator.appName;
  // ブラウザのバージョンを表示
  ele[1].innerHTML = navigator.appVersion;
  // ユーザーエージェントを表示
  ele[2].innerHTML = navigator.userAgent;
}, false);
```

index.htmlをIE9で表示すると、次のように、ブラウザに関する情報が表示されます。



Google Chromeでは、次のように、ブラウザに関する情報が表示されます。



ONEPOINT **ブラウザに関する情報を取得するには「navigator」オブジェクトを使う**

「navigator」オブジェクトには、ブラウザに関する情報が格納されています。アプリケーション名(ブラウザ名)やバージョンなどを取得できます。ただし、過去におけるブラウザの遺産ともいえますが、実際には正しい情報は得られません。ブラウザを判別する方法については、下記のCOLUMNを参照してください。

「navigator」オブジェクトのプロパティは、次の表のようになります。

プロパティ	機能
appCodeName	ブラウザのコード名を返す
appName	ブラウザ名を返す
appVersion	ブラウザのバージョン名を返す
userAgent	ユーザエージェントを返す

COLUMN **ブラウザを判別するには**

ブラウザを判別するのであれば、「userAgent」プロパティに格納された文字列から判別する必要があります。ただし、この「userAgent」プロパティもブラウザによって切り替えたり、変更したりすることができるため、確実に特定のブラウザであると判断できる材料にはなりません。

どうしてもブラウザを特定しなければならない場合は、そのブラウザだけに実装されているオブジェクトやプロパティがあるかどうかを調べた方が確実性が高くなります。たとえば、IEであれば、「document.uniqueID」のようにして「uniqueID」プロパティがあるかどうかで判別ができます。Operaの場合は、「window.opera」として調べることができます。

- **ユーザーエージェント一覧**

URL http://www.openspc2.org/userAgent/

SECTION-104

JavaやGeolocationなどの機能が使えるかどうか調べる

ここでは、JavaやGeolocationなどの機能が使えるかどうか調べる方法について解説します。

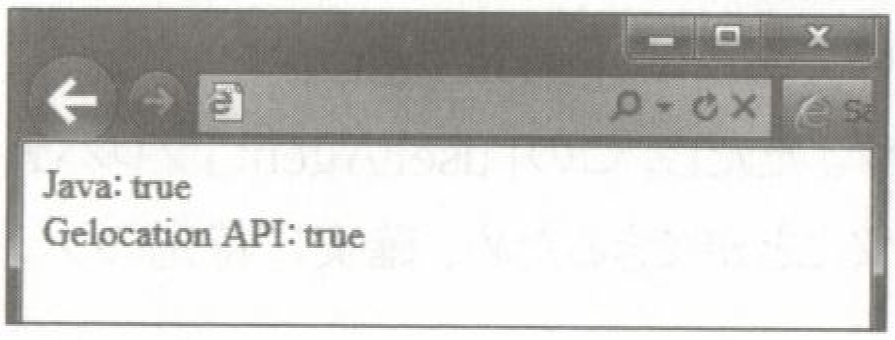
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Javaが使えるかどうか調べる
  var flag1 = navigator.javaEnabled();
  // Geolocation APIが使えるかどうか調べる
  var flag2 = navigator.geolocation ? true: false;
  // 結果を出力
  ele.innerHTML = "Java:"+flag1+"<br>";
  ele.innerHTML += "Gelocation API:"+flag2+"<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、JavaとGeolocationが使用できるかどうかの結果が表示されます。



ONEPOINT 利用できる機能のチェックを行うにはオブジェクトの存在を確認する

JavaScriptでは利用可能な機能があるかどうか調べる際、オブジェクトが存在するかどうかで判断する場合があります。たとえば、Geolocation APIの場合、「navigator」オブジェクト内に「geolocation」オブジェクトがあるかどうかで判断できます。なお、Javaが使用できるかどうかに関しては、「javaEnabled()」メソッドを使って調べることができます。

接続回線の種類を調べる

ここでは、Wi-Fiや3G回線など、端末が接続されている回線の種類を調べる方法について解説します。

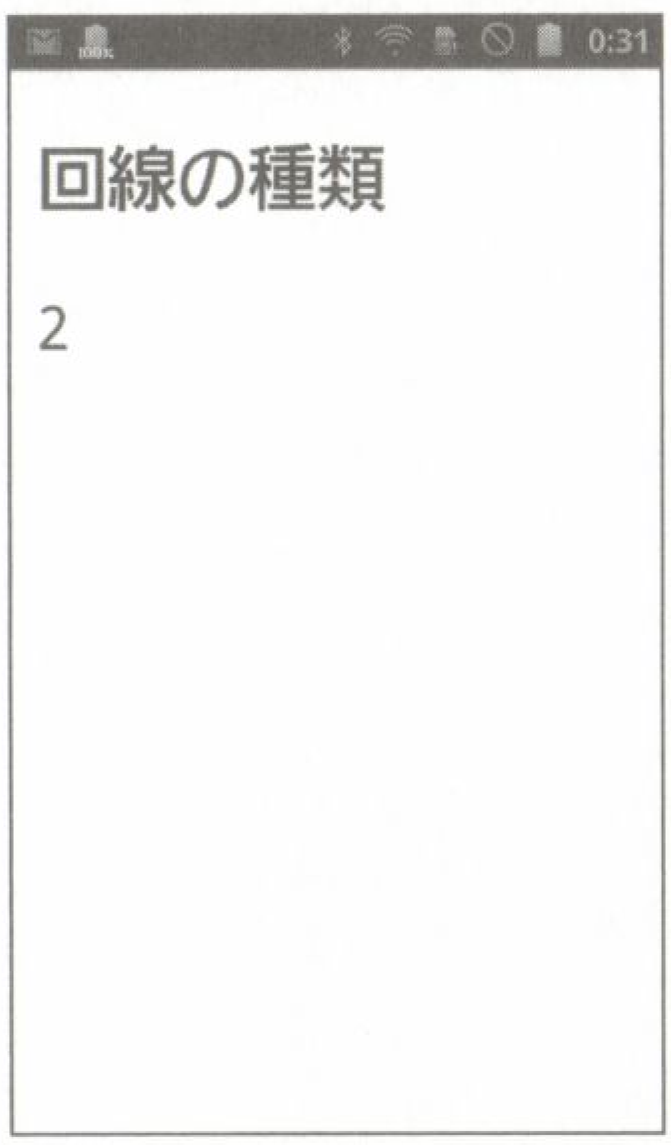
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>回線の種類</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 回線の種類を調べる
  var lineType = navigator.connection.type;
  // 結果を出力
  ele.innerHTML = lineType;
}, false);
```

ブラウザでindex.htmlを表示すると、回線の種類を示す値が表示されます。



ONEPOINT **接続回線の種類を調べるには「navigator.connection.type」プロパティを使う**

Android 2.2以降では、「navigator.connection.type」プロパティで回線の種類を調べることができます。Wi-Fiか3G通信かを調べることができるので、回線の種類によって表示する映像などを切り換えることもできます。なお、Network Information APIの仕様とは若干、違いがあり、「navigator.connection.type」プロパティは文字列でなく、次の表に示す値を返します。

値	回線の種類
0	不明
1	イーサネット
2	Wi-Fi
3	2G通信
4	3G通信

Network Information APIについては、次のURLを参照してください。

- Network Information API

URL http://www.w3.org/TR/netinfo-api/

IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-106

ヒストリー数を求める

ここでは、ヒストリー数を求める方法について解説します。

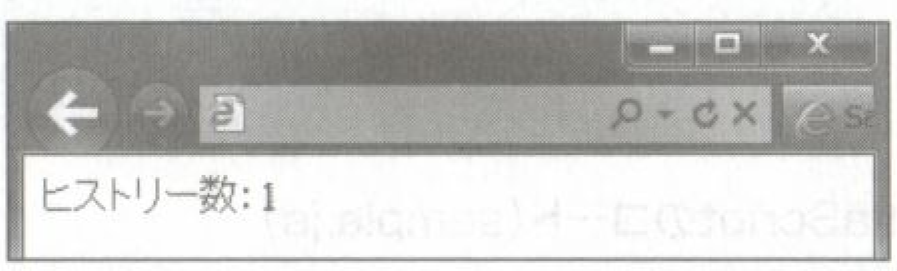
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ヒストリーの数を調べる
  var len = history.length;
  // 結果を出力
  ele.innerHTML = "ヒストリー数:"+len;
}, false);
```

ブラウザでindex.htmlを表示すると、ヒストリー数が表示されます。



ONEPOINT ヒストリー数を取得するには「history」オブジェクトの「length」プロパティを使う

ヒストリー数は、「history」オブジェクトの「length」プロパティを参照することで求めることができます。ヒストリーは、ページの閲覧を進めたり、戻したりすることで変化します。ヒストリー数は、「back()」メソッドや「forward()」メソッド、「go()」メソッドでも変化します。

また、「pushState()」メソッドと「replaceState()」メソッドを使えるブラウザでは、プログラムによってヒストリーを制御することもできます。

関連項目 ▶▶▶

●ページを進める/戻す ……………… p.256
●ヒストリーを追加・置換する ……………… p.258

SECTION-107

ページを進める/戻す

ここでは、ページを進める/戻す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

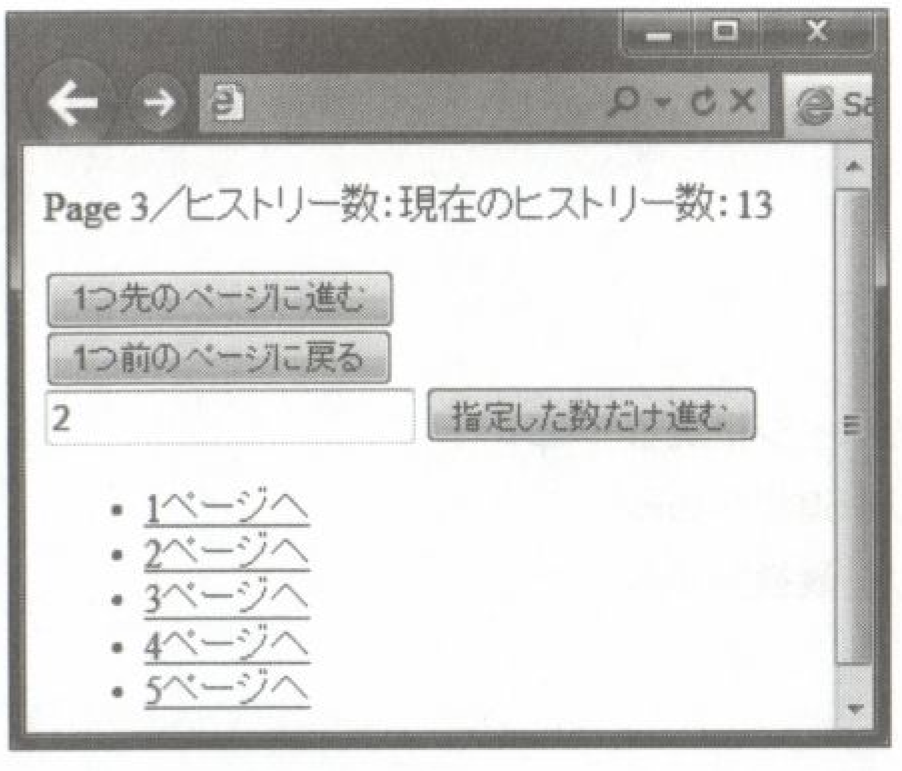
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p>Page 1／ヒストリー数：<output>0</output></p>
    <form>
      <input type="button" id="nextHis" value="1つ先のページに進む"><br>
      <input type="button" id="prevHist" value="1つ前のページに戻る"><br>
      <input type="text" value="2" id="count">
      <input type="button" id="goHist" value="指定した数だけ進む"><br>
    </form>
    <ul>
      <li><a href="index.html">1ページへ</a>
      <li><a href="index2.html">2ページへ</a>
      <li><a href="index3.html">3ページへ</a>
      <li><a href="index4.html">4ページへ</a>
      <li><a href="index5.html">5ページへ</a>
    </ul>
  </body>
</html>
```

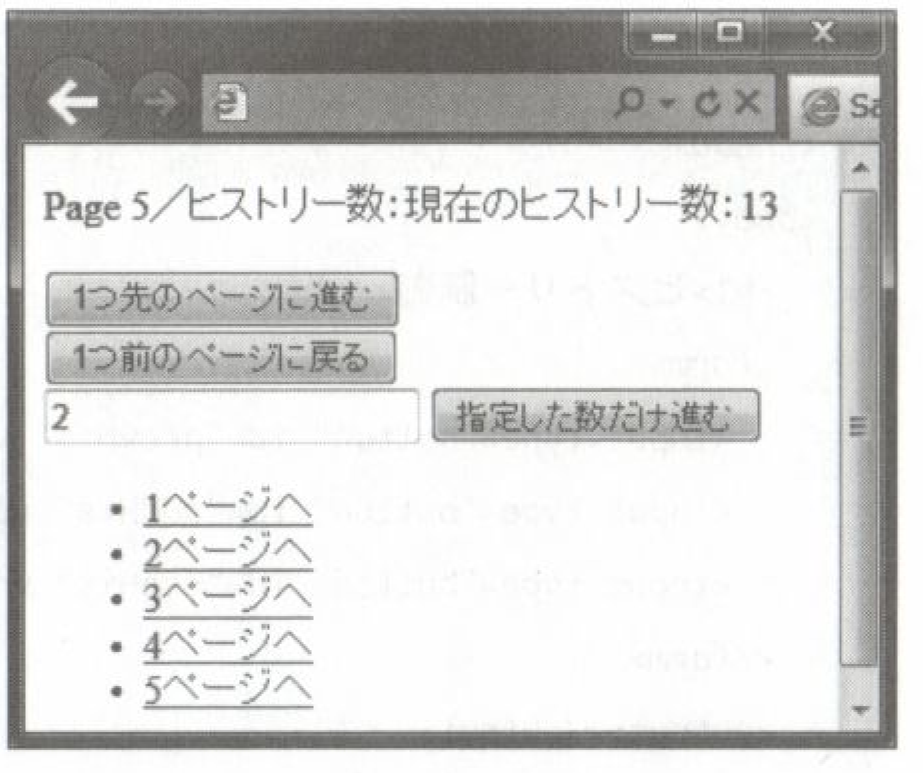
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var btn1 = document.getElementById("nextHis");
  var btn2 = document.getElementById("prevHist");
  var btn3 = document.getElementById("goHist");
  // 次のヒストリーに進む
  btn1.addEventListener("click", function(){
    history.forward();
  }, false);
  // 前のヒストリーに戻る
  btn2.addEventListener("click", function(){
    history.back();
  }, false);
```

```
  // 指定した数だけヒストリーを進める
  btn3.addEventListener("click", function(){
    var n = document.getElementById("count").value;
    history.go(n);
  }, false);
  // 結果を出力
  ele.innerHTML = "現在のヒストリー数 :"+history.length;
}, false);
```

ブラウザでindex.htmlを表示し、「1ページ」～「5ページ」のリンクを数回、クリックした後で、ボタンをクリックすると、クリックしたボタンに応じてページが移動します。





ONEPOINT **ヒストリー間でページを移動するには「back()」「forward()」「go()」メソッドを使う**

ヒストリーに記録された範囲であれば、「back()」メソッドや「forward()」メソッド、「go()」メソッドを使って自由にページを移動することができます。「back()」メソッドは、1つ前のヒストリーに記録されたページに戻ります。その際、ヒストリーがない場合でもエラーにはなりません。「forward()」メソッドは、1つ先のヒストリーに記録されたページに移動します。進むべきページが記録されていない場合でもエラーにはなりません。「back()」メソッド、「forward()」メソッドとも、パラメータはありません。

ヒストリーを任意数だけ移動する場合は、「go()」メソッドを使います。「go()」メソッドのパラメータには、移動するヒストリー数を指定します。正数であれば先のページに進み、負数であれば前のページに戻ります。「back()」メソッドと「forward()」メソッドと同様に、戻るべきページや進むべきページがない場合でもエラーにはなりません。

関連項目 ▶▶▶

● ヒストリー数を求める …… p.255
● ヒストリーを追加・置換する …… p.258

SECTION-108

ヒストリーを追加・置換する

ここでは、ヒストリーを追加・置換する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ヒストリー制御</h1>
    <form>
      <input type="button" id="prevHis" value="1つ前のページに戻る"><br>
      <input type="button" id="addHis" value="ヒストリー追加"><br>
      <input type="button" id="repHis" value="ヒストリー置換"><br>
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

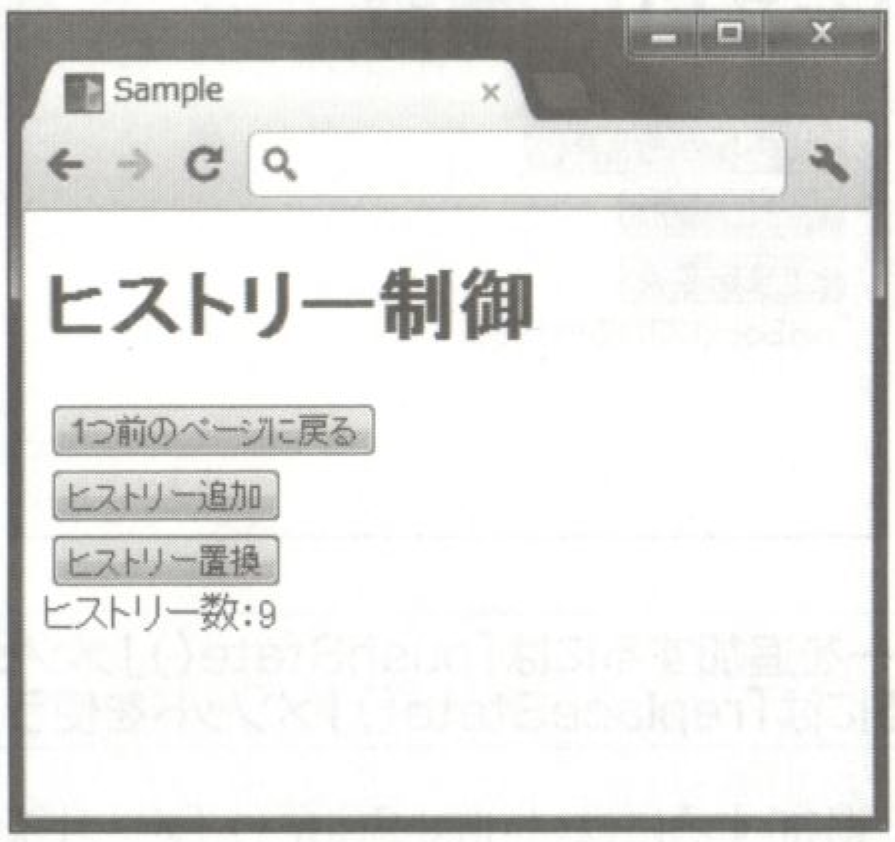
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // windowにヒストリー変更時のイベントを設定
  window.addEventListener("popstate", function(evt){
    // pushStateで指定されたステートを表示
    ele.innerHTML = event.state;
  }, false);
  // ボタン要素を読み出し
  var btn1 = document.getElementById("prevHis");
  var btn2 = document.getElementById("addHis");
  var btn3 = document.getElementById("repHis");
  // ボタンにイベントを割り当て
  btn1.addEventListener("click", function(){
    // 前のヒストリーに戻る
    history.back();
  }, false);
  // ヒストリーを追加
  btn2.addEventListener("click", function(){
    var t = (new Date()).getTime();
    history.pushState("newBook"+t, "新刊", "new.html");
```

```
    ele.innerHTML = "ヒストリー数:"+history.length;
  }, false);
  // ヒストリーを置換
  btn3.addEventListener("click", function(){
    var t = (new Date()).getTime();
    history.replaceState("oldBook"+t, "古書", "old.html");
    ele.innerHTML = "ヒストリーを置換しました";
  }, false);
}, false);
```

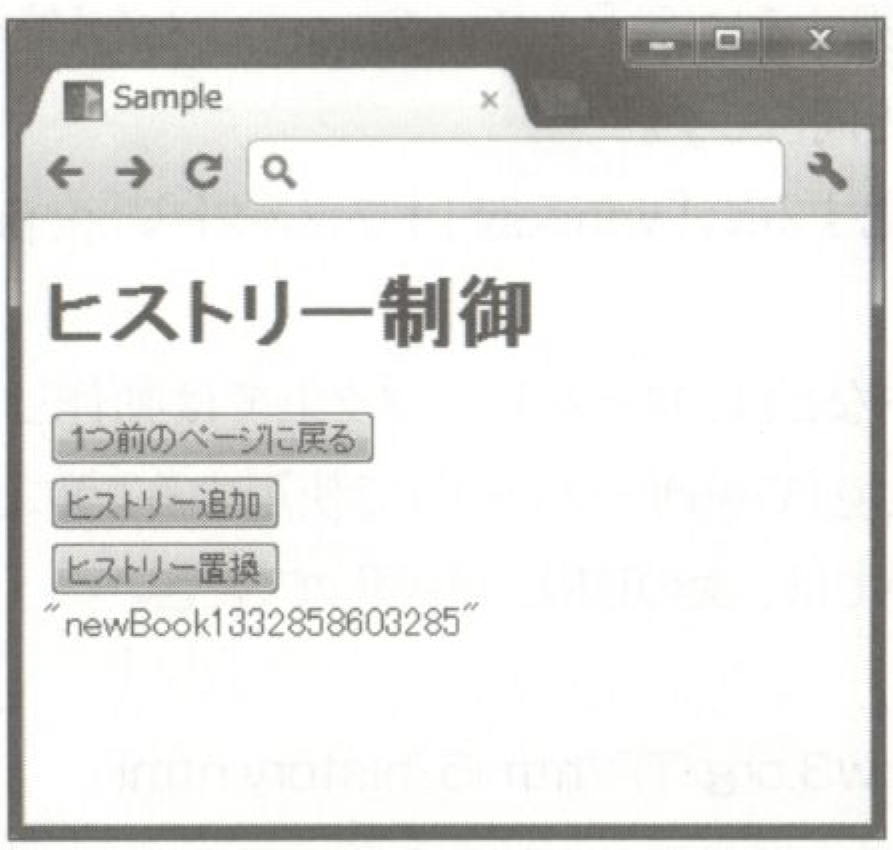
HINT

このサンプルは、ローカルディスク上では動作しないブラウザもあります。その場合は、ネットワーク環境（Webサーバー上）で動作を確認してください。

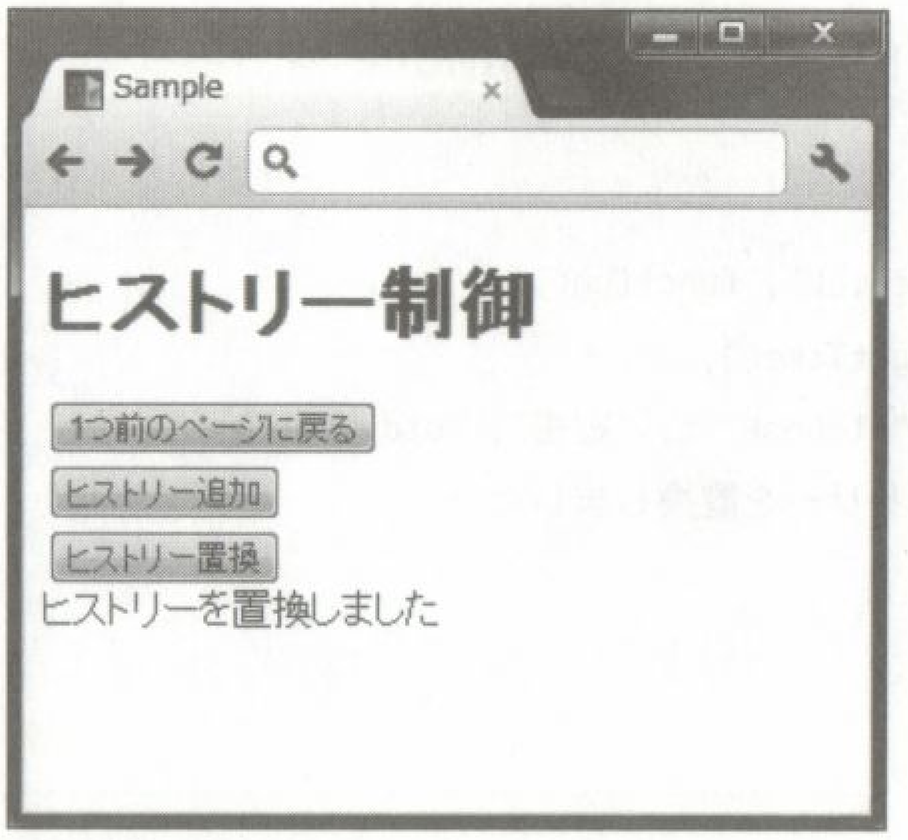
ブラウザでindex.htmlを表示し、「ヒストリー追加」ボタンをクリックすると、ヒストリーが追加されます。



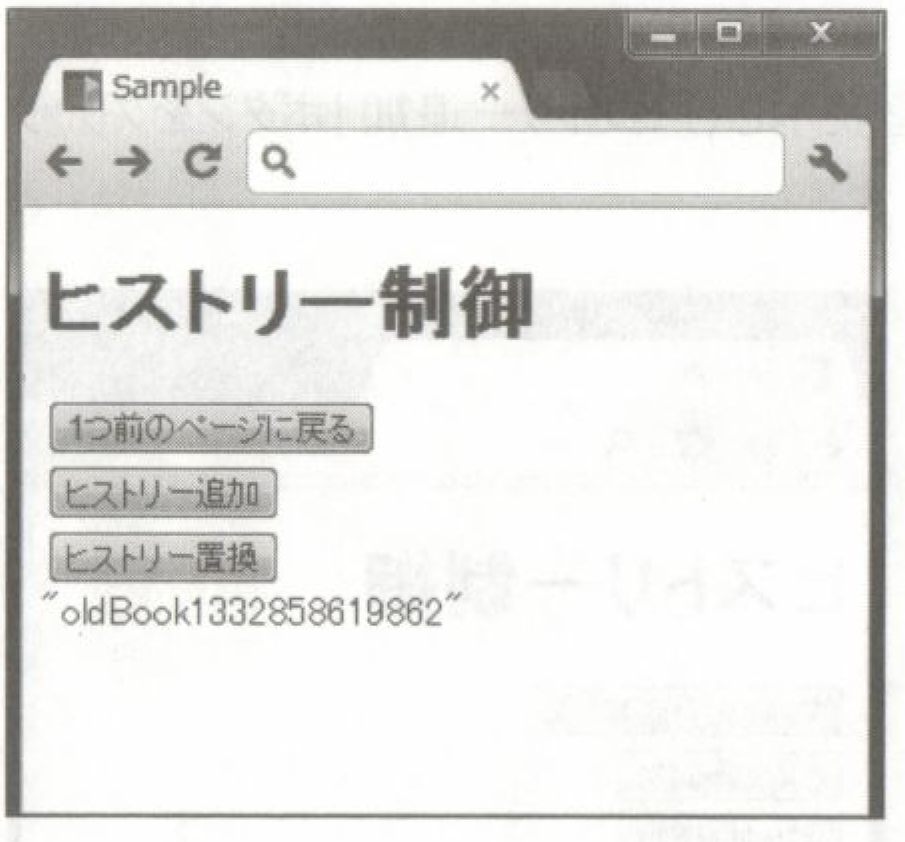
「1つ前のページに戻る」ボタンをクリックすると、プログラムで設定したヒストリーの情報（文字）が追加されます。



「ヒストリー置換」ボタンをクリックすると、現在のヒストリーが置換されます。



ヒストリーを移動すると、置換されていることが確認できます。



ONEPOINT **ヒストリーを追加するには「pushState()」メソッド、置換するには「replaceState()」メソッドを使う**

スクリプトでヒストリーを追加するには、「pushState()」メソッドを使います。パラメータは3つあり、最初のパラメータがイベント発生時に渡される「state」プロパティに入るデータになります。2番目のデータがヒストリーのタイトル(ページタイトル)、3番目がURLになります。

現在のヒストリーを置換するには、「replaceState()」メソッドを使います。パラメータは、「pushState()」メソッドと同じです。

また、ヒストリーが変化すると、「window」オブジェクトで「popstate」イベントが発生します。

なお、これらのメソッドなどは、ローカルディスク上では動作しないブラウザもあります。このため、ネットワーク環境(Webサーバー上)で使用するようにしてください。

Histrory APIについては、次のURLを参照してください。

- Histrory API

  URL http://www.w3.org/TR/html5/history.html

関連項目 ▶▶▶

- ヒストリー数を求める ………… p.255
- ページを進める/戻す ………… p.256

IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-109

画面のサイズを求める

ここでは、画面のサイズを求める方法について解説します。

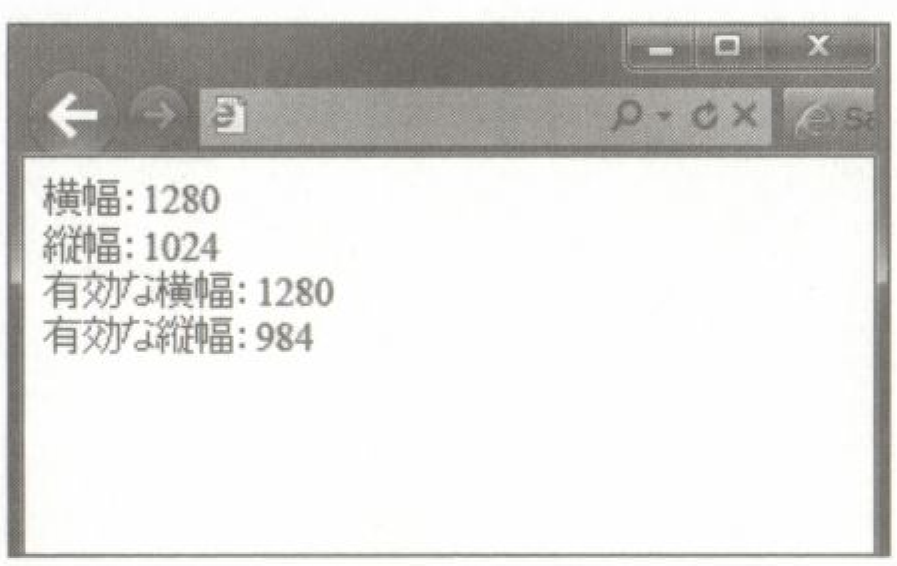
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 画面の横幅を取得
  var w = screen.width;
  // 画面の縦幅を取得
  var h = screen.height;
  // 画面の有効な横幅を取得
  var aw = screen.availWidth;
  // 画面の有効な縦幅を取得
  var ah = screen.availHeight;
  // 結果を出力
  ele.innerHTML = "横幅 :"+w+"<br>";
  ele.innerHTML += "縦幅 :"+h+"<br>";
  ele.innerHTML += "有効な横幅 :"+aw+"<br>";
  ele.innerHTML += "有効な縦幅 :"+ah;
}, false);
```

ブラウザでindex.htmlを表示すると、画面のサイズと実際に利用できる画面のサイズが表示されます。



ONEPOINT **モニタ画面のサイズを調べるには「screen」オブジェクトの「width」プロパティや「height」プロパティを使う**

モニタ画面のサイズは、「screen」オブジェクトの「width」プロパティと「height」プロパティで調べることができます。また、メニューバーなどを除いた実際に使用できる画面サイズは、「availWidth」プロパティと「availHeight」プロパティで調べることができます。

COLUMN **マルチモニタの場合**

モニタ画面が複数ある場合(マルチモニタやデュアルモニタなど)は、ブラウザによって動作が異なります。

IE9、Safari 5~6、Opera 12(Mac OS X/Windows 7)では、メニューバーやタスクバーがあるメイン画面のサイズが「width」プロパティと「height」プロパティに入ります。複数の画面のすべての幅を加算したピクセル数にはなりません。また、第二画面など、他の画面にブラウザウィンドウを移動させた状態で「width」プロパティと「height」プロパティを読み出しても、メニューバーやタスクバーがあるメイン画面の幅しか取得することができません。

Google Chrome 21、Firefox 14、Safari 4では、ウィンドウが表示されている面積が大きいモニタ画面のサイズが「width」プロパティと「height」プロパティに入ります。この場合でも、すべてのモニタ画面の合計ピクセル数にはなりません。

「width」プロパティと「height」プロパティはブラウザのバージョンによって異なる結果を返すので、マルチモニタを前提とする場合は注意が必要です。

関連項目 ▶▶▶
- 画面の色深度を求める ……………… p.263
- 画面の解像度を調べる ……………… p.265
- ウィンドウの幅を求める ……………… p.267

画面の色深度を求める

ここでは、画面の色深度を求める方法について解説します。

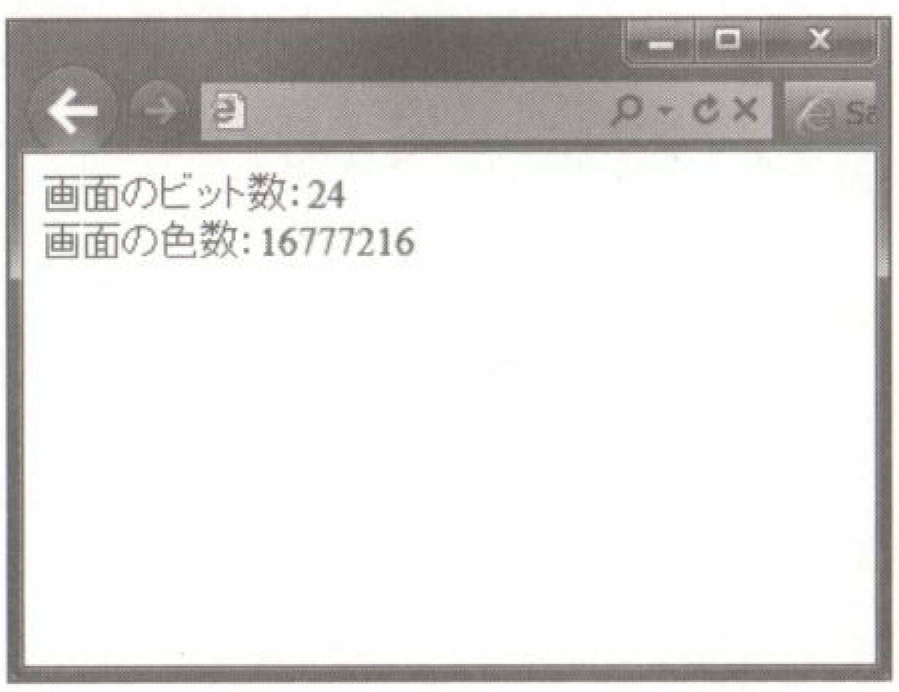
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 画面の色深度を調べる
  var bit = screen.colorDepth;
  // ビット数から画面の色数を求める
  var n = Math.pow(2, bit);
  // 結果を出力
  ele.innerHTML = "画面のビット数："+bit+"<br>";
  ele.innerHTML += "画面の色数："+n;
}, false);
```

ブラウザでindex.htmlを表示すると、色深度のビット数と、最大表示色数が表示されます。



ONEPOINT 画面の色深度を調べるには「screen」オブジェクトの「colorDepth」プロパティを使う

画面に表示される文字やグラフィックの色は「赤」(red)、「青」(Blue)、「緑」(Green)の三原色の組み合わせです。コンピュータでは、1色に付き何ビット割り当てるかで表現できる色数が変わります。多くの環境では、フルカラーと呼ばれるRGB各色8ビットが利用されます(OSによっては内部では16ビットの場合もある)。このビット数がどのくらいあるかを調べるには、「screen」オブジェクトの「colorDepth」プロパティを使います。結果はビット数になるので、色数を求めたい場合はサンプルのように、「Math.pow()」メソッドを利用します。

ビット数と色数は、次の表のようになります。24ビットと32ビットは、同じ色数になります。

ビット数	色数
1	白黒
2	4色
3	8色
4	16色
8	256色
12	4096色
16	32768色
24	16777216色
32	16777216色

なお、Operaでは、フルカラーの場合、「colorDepth」プロパティは「32」を返します。

関連項目 ▶▶▶

- 画面のサイズを求める …… p.261
- 画面の解像度を調べる …… p.265
- ウィンドウの幅を求める …… p.267

画面の解像度を調べる

ここでは、画面の解像度を調べる方法について解説します。

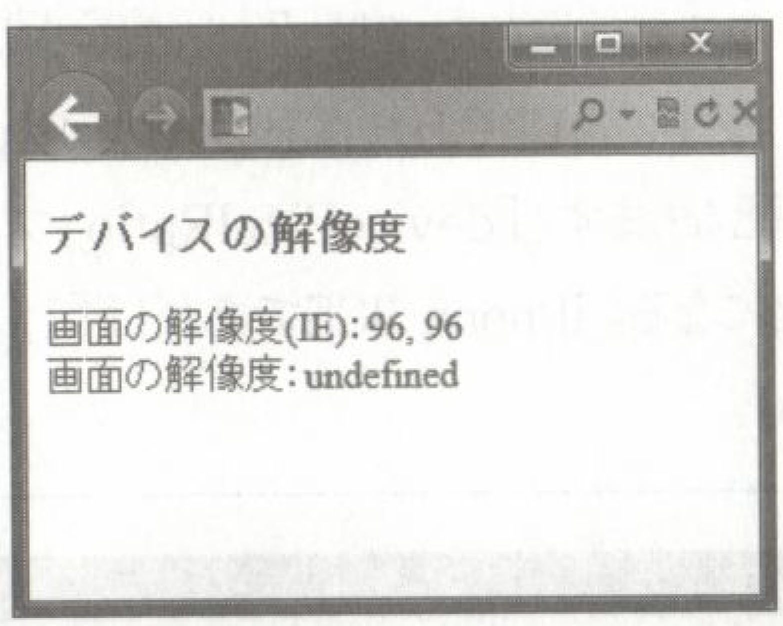
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>デバイスの解像度</h3>
    <output></output>
  </body>
</html>
```

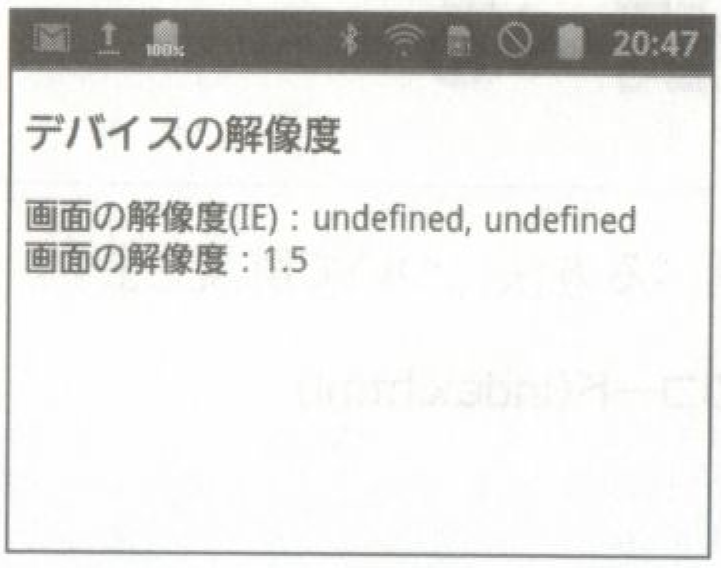
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // IEで画面の解像度を調べる
  var xdpi = screen.deviceXDPI;
  var ydpi = screen.deviceYDPI;
  // 結果を出力
  ele.innerHTML = "画面の解像度(IE):"+xdpi+", "+ydpi+"<br>";
  // Androidなどの場合
  var dpi = window.devicePixelRatio;
  // 結果を出力
  ele.innerHTML += "画面の解像度 :"+dpi;
}, false);
```

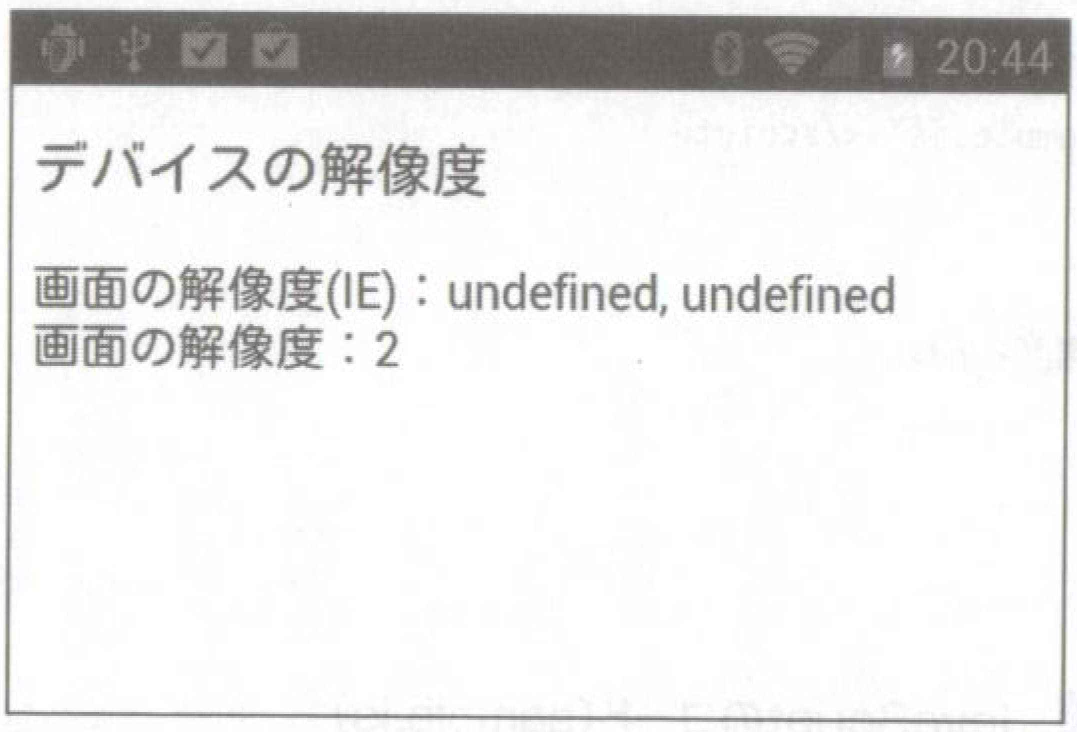
index.htmlを表示すると、IEでは画面の解像度が表示されます。



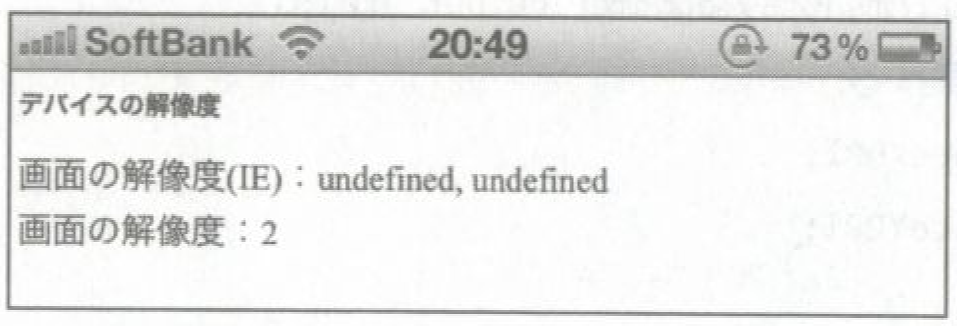
Galaxy SII（Android 2.3）では、1.5倍の解像度であることが表示されます。



Galaxy Nexus（Android 4）では、2倍の解像度であることが表示されます。



iPhone 4S（iOS 5）では、2倍の解像度であることが表示されます。



ONEPOINT **画面の解像度を調べるには「deviceXDPI」プロパティや「deviceYDPI」プロパティ、「devicePixelRatio」プロパティを使う**

画面の解像度は、96dpi（dpi=dot per inch）が基準になっています。ただし、デバイスによっては、より高い解像度の場合もあります。この解像度を調べるには、IEの場合は「screen」オブジェクトの「deviceXDPI」プロパティや「deviceYDPI」プロパティを参照します。これらのプロパティは、96dpiであれば96の数値を返します。

iPhoneやAndroidなどのスマートフォンでは、「window」オブジェクトの「devicePixelRatio」プロパティを参照します。「1」であれば1ピクセルが1px、「2」であれば1ピクセルが画面では2px分の描画となります（「devicePixelRatio」プロパティの値が「2」であれば、2倍の細かさということになる。iPhone 3GSであれば「1」、2倍の画面の細かさであるiPhone 4では「2」となる）。

関連項目 ▶▶▶

● 画面のサイズを求める ………… p.261
● 画面の色深度を求める ………… p.263
● ウィンドウの幅を求める ………… p.267

ウィンドウの幅を求める

ここでは、ウィンドウの幅を求める方法について解説します。

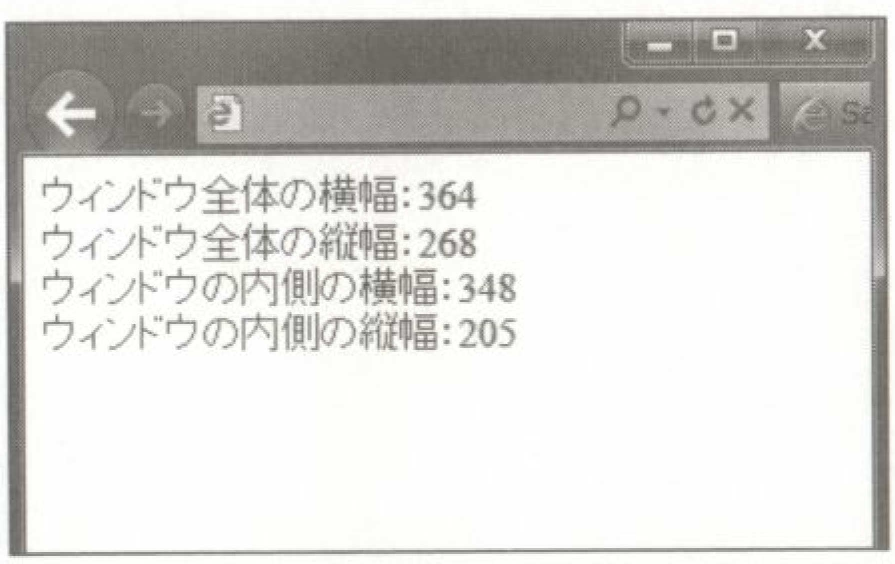
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ウィンドウの横幅を取得
  var w = window.outerWidth;
  // ウィンドウの縦幅を取得
  var h = window.outerHeight;
  // ウィンドウの有効な横幅を取得
  var iw = window.innerWidth;
  // ウィンドウの有効な縦幅を取得
  var ih = window.innerHeight;
  // 結果を出力
  ele.innerHTML = "ウィンドウ全体の横幅 :"+w+"<br>";
  ele.innerHTML += "ウィンドウ全体の縦幅 :"+h+"<br>";
  ele.innerHTML += "ウィンドウの内側の横幅 :"+iw+"<br>";
  ele.innerHTML += "ウィンドウの内側の縦幅 :"+ih;
}, false);
```

ブラウザでindex.htmlを表示すると、ウィンドウの外側の幅と内側の幅のサイズが表示されます。



ONEPOINT

ウィンドウ幅を求めるには「outerWidth」「outerHeight」「innerWidth」「innerHeight」プロパティを使う

ウィンドウにはタイトルバーやクローズボタンなどがあり、これらを含めたサイズを求める場合は、「window」オブジェクトの「outerWidth」プロパティと「outerHeight」プロパティを参照します。なお、ウィンドウの影は、「outerWidth」プロパティと「outerHeight」プロパティには含まれません。

ウィンドウの外側ではなく、実際にWebページが表示される部分のサイズは、「innerWidth」プロパティと「innerHeight」プロパティを参照します。

いずれのプロパティも得られた値の単位は、ピクセルになります。

関連項目 ▶▶▶

- 画面のサイズを求める …………………… p.261
- 画面の色深度を求める …………………… p.263
- 画面の解像度を調べる …………………… p.265

指定位置までスクロールさせる

ここでは、指定位置までスクロールさせる方法について解説します。

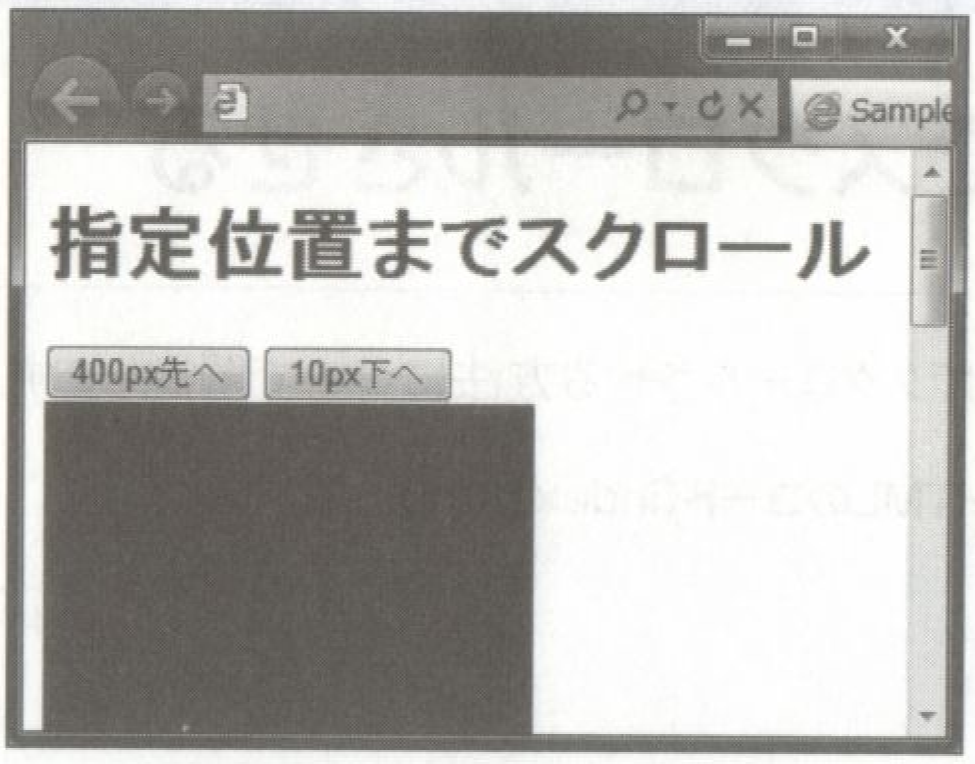
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>指定位置までスクロール</h1>
    <form>
      <input type="button" value="400px先へ" id="to400">
      <input type="button" value="10px下へ" id="by10">
    </form>
    <img src="images/photo.jpg" width="200" height="750">
  </body>
</html>
```

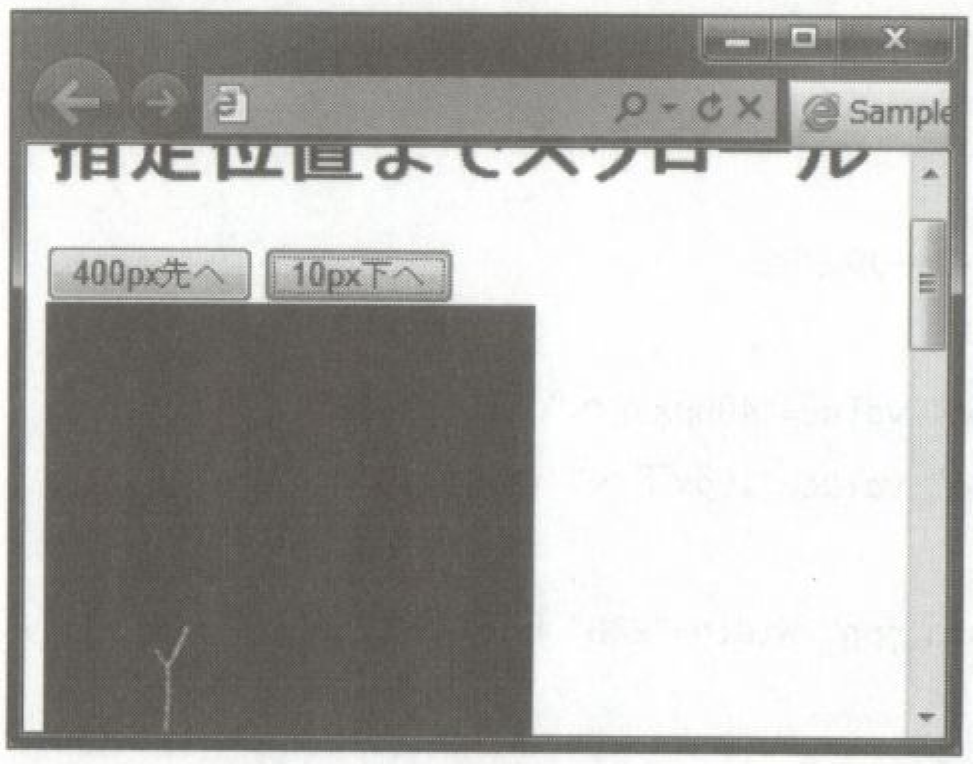
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn1 = document.getElementById("to400");
  var btn2 = document.getElementById("by10");
  // ボタンにイベントを割り当て
  btn1.addEventListener("click", function(){
    // 指定した位置(400px先)までスクロールする
    window.scrollTo(0, 400);
  }, false);
  // ボタンにイベントを割り当て
  btn2.addEventListener("click", function(){
    // 指定した量(10px)だけスクロールする
    window.scrollBy(0, 10);
  }, false);
}, false);
```

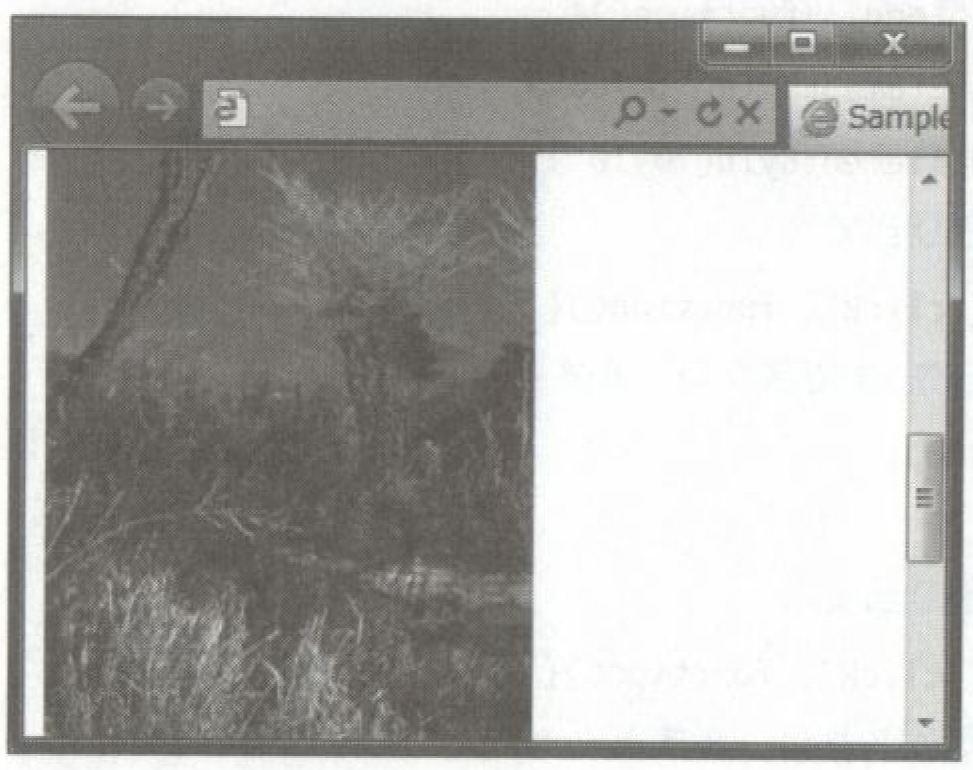
ブラウザでindex.htmlを表示し、ボタンをクリックすると、指定した位置にスクロールするか指定した量だけスクロールします。



「scrollBy()」メソッドの場合、相対的にスクロールします。



「scrollTo()」メソッドの場合、ページ左上を原点として指定したX座標とY座標までスクロールします。



ONEPOINT

ページ内容をスクロールさせるには「scrollTo()」メソッドや「scrollBy()」メソッドを使う

ページ内容をスクロールさせるには、「scrollTo()」メソッド、または、「scrollBy()」メソッドを使います。「scrollTo()」メソッドは、指定したX座標とY座標までページをスクロールさせます。「scrollBy()」メソッドは、指定した移動量だけページをスクロールさせます。

「scrollTo()」メソッドと「scrollBy()」メソッドとも、瞬時にスクロールします(Windows Phoneを除く)。スムーズにスクロールさせる場合は、タイマーを使って処理するか、スムーズスクロールするライブラリやプラグインを使用する必要があります。

COLUMN スマートフォンでアドレスバーを隠す

iPhoneやAndroidなどのスマートフォンでは、ページの読み込みが完了した後、数ミリ秒後にスクロールさせることでアドレスバーを自動的に隠すことができます。なお、ページがスクロール可能でない場合は、アドレスバーを隠すことはできません。

```
window.addEventListener("load", function(){
  setTimeout("scrollTo(0,1)", 100);
}, false);
```

関連項目 ▶▶▶

● スクロール時に処理を行う …… p.431

SECTION-114

Base64エンコード/デコードする

ここでは、Base64エンコード/デコードする方法について解説します。

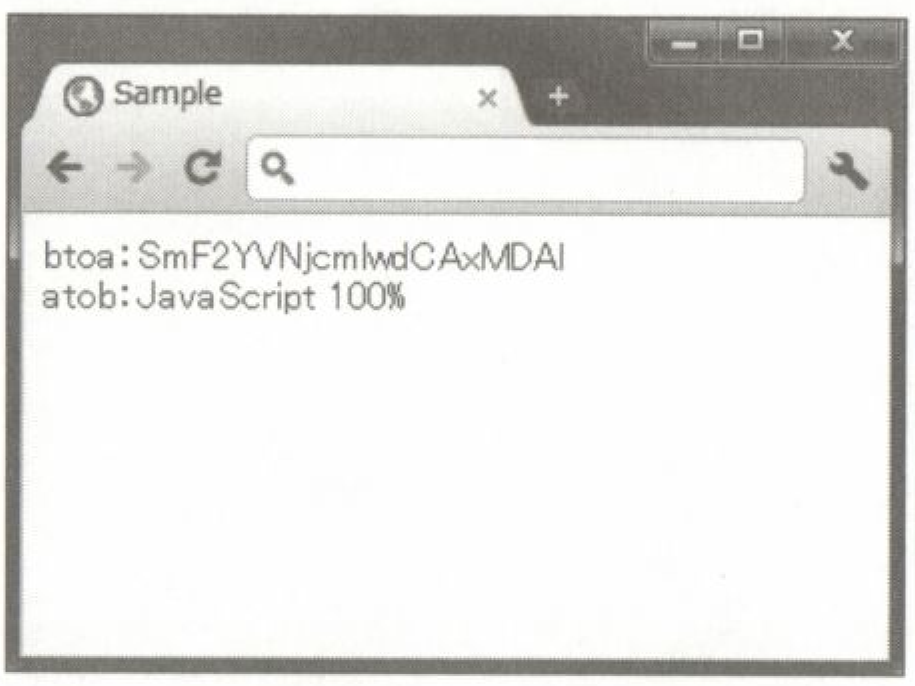
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // エンコードした結果を代入
  var result1 = window.btoa("JavaScript 100%");
  // デコードした結果を代入
  var result2 = window.atob(result1);
  // 結果を出力
  ele.innerHTML = "btoa:"+result1+"<br>";
  ele.innerHTML += "atob:"+result2;
}, false);
```

ブラウザでindex.htmlを表示すると、文字列をBase64エンコードした結果と、デコードした結果が表示されます。



ONEPOINT **Base64エンコードするには「btoa()」メソッド、Base64デコードするには「atob()」メソッドを使う**

バイナリデータなどをBase64データ(64種類の文字列)として変換(エンコード)する場合には、「btoa()」メソッドを使います。パラメータに指定されたデータは、Base64形式で示される文字列に変換されます。Base64エンコードされた文字列を元に復元(デコード)するには、「atob()」メソッドを使います。

Base64で使用される文字は、次の表のようになります。また、不足ビット数を埋める文字として「=」が使用されます。

値	対応文字	値	対応文字	値	対応文字
0	A	22	W	43	r
1	B	23	X	44	s
2	C	24	Y	45	t
3	D	25	Z	46	u
4	E	26	a	47	v
5	F	27	b	48	w
6	G	28	c	49	x
7	H	29	d	50	y
8	I	30	e	51	z
9	J	31	f	52	0
10	K	32	g	53	1
11	L	33	h	54	2
12	M	34	i	55	3
13	N	35	j	56	4
14	O	36	k	57	5
15	P	37	l	58	6
16	Q	38	m	59	7
17	R	39	n	60	8
18	S	40	o	61	9
19	T	41	p	62	+
20	U	42	q	63	/
21	V				

なお、Base64については、次のURLを参照してください。

- Base64

URL http://tools.ietf.org/html/rfc3548

SECTION-115

印刷する

ここでは、ページ内容を印刷する方法について解説します。

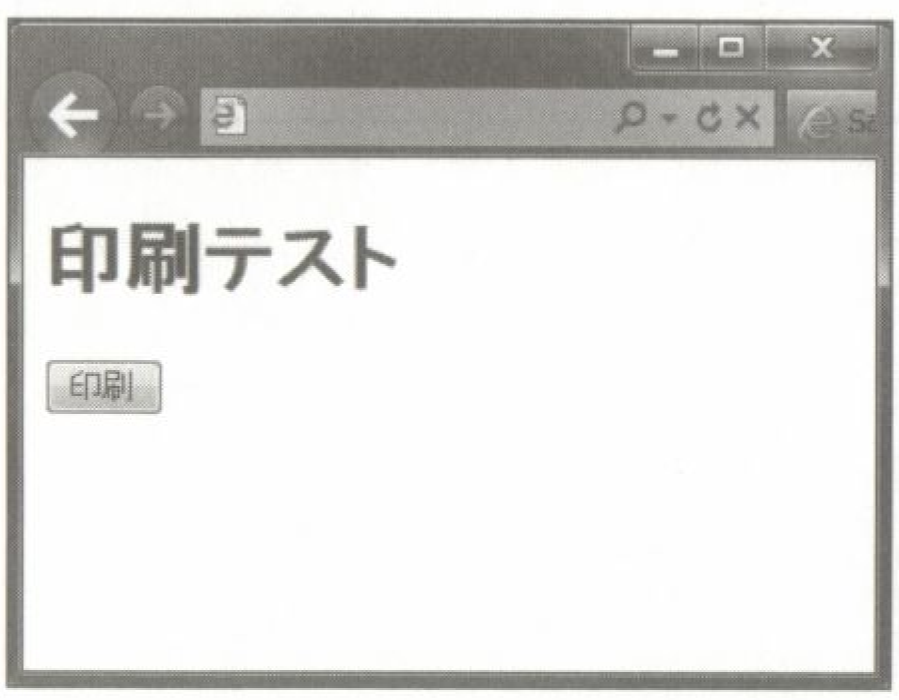
SAMPLE CODE HTMLのコード(index.html)

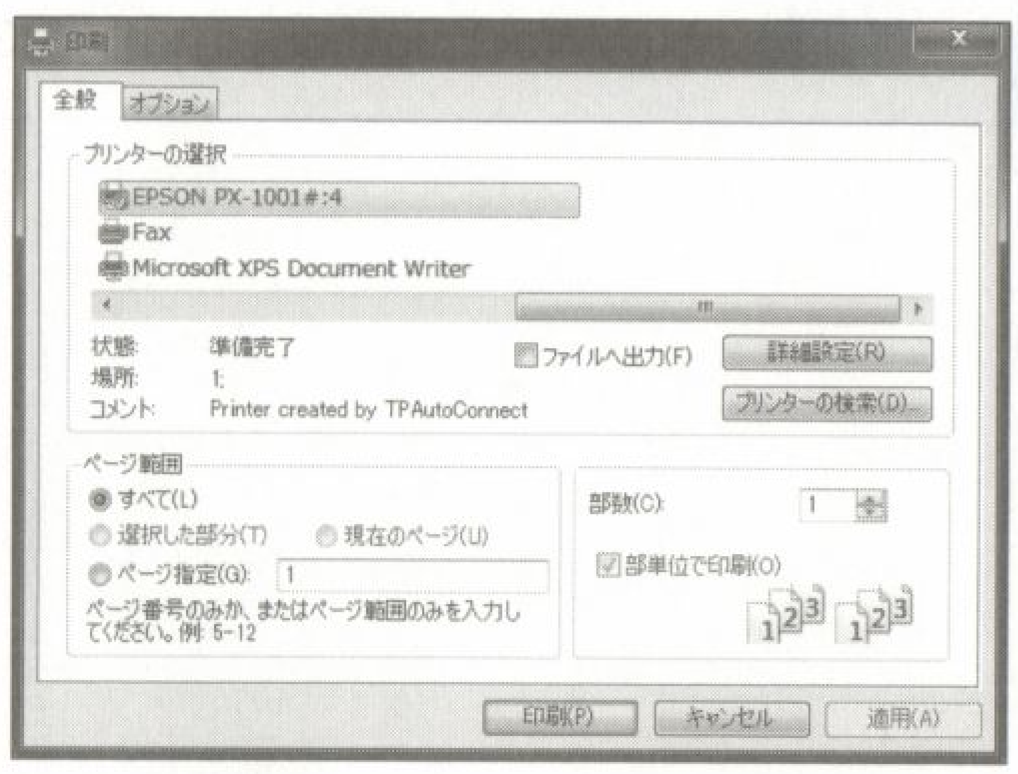
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>印刷テスト</h1>
    <form>
      <input type="button" value="印刷">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("input")[0];
  // ボタンにイベントを割り当て
  ele.addEventListener("click", function(){
    // 印刷する
    window.print();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、印刷ダイアログが表示され、ページ内容が印刷されます。





ONEPOINT 印刷するには「print()」メソッドを使う

ページ内容を印刷するには、「print()」メソッドを使います。ウィンドウだけでなく、フレームを指定して印刷することもできます。また、iOSでは、Air Printに対応している必要があります。Android 2～4では、「print()」メソッドを実行しても何も起こりません。

印刷内容を変更する場合には、「beforeprint」イベントと「afterprint」イベントを利用します。イベントハンドラ内では、CSSなどを使って、印刷したくないものは非表示にしておきます。

SECTION-116

選択された文字を取得する

ここでは、選択された文字を取得する方法について解説します。

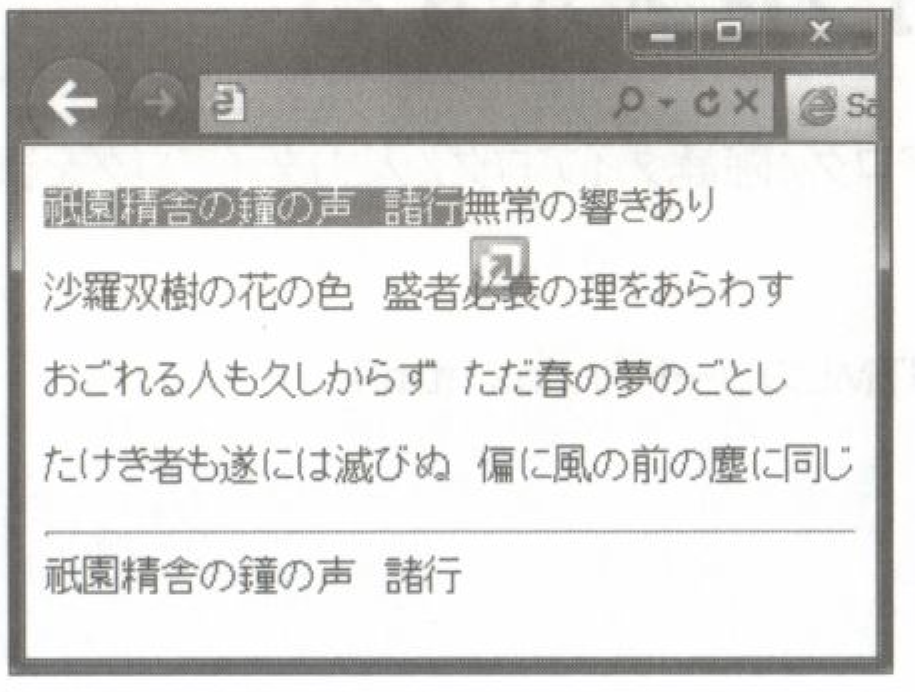
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <div>
      <p>祇園精舎の鐘の声　諸行無常の響きあり</p>
      <p>沙羅双樹の花の色　盛者必衰の理をあらわす</p>
      <p>おごれる人も久しからず　ただ春の夢のごとし</p>
      <p>たけき者も遂には滅びぬ　偏に風の前の塵に同じ</p>
      <hr>
      <output></output>
    </div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // タイマーを使って選択された文字を定期的に表示
  setInterval(function(){
    // 選択範囲を保持するオブジェクトを取得
    var selObj = window.getSelection();
    // 最初に選択された選択範囲から文字を取得
    var text = selObj.getRangeAt(0);
    // 選択された文字を出力
    ele.innerHTML = text;
  }, 1000);
}, false);
```

ブラウザでindex.htmlを表示し、ページ内の文字を選択すると、選択した文字が画面にほぼリアルタイムに表示されます。



ONEPOINT

選択された文字を取得するには「getSelection()」メソッドと「getRangeAt()」メソッドを使う

ウィンドウ上で選択された文字を取得するには、「getSelection()」メソッドを使って選択範囲の情報を持つ「Selection」オブジェクトを取得します。選択されたテキストは「getRangeAt()」メソッドを使って読み出すことができます。Firefoxの場合、複数の選択範囲を持つことができるため、「getRangeAt()」のパラメータは「0」だけでなく、「1」以上の場合もあります。選択範囲がいくつあるかは、「Selection」オブジェクトの「rangeCount」プロパティで求めることができます。

SECTION-117

アラートダイアログ/確認ダイアログ/入力ダイアログを表示する

ここでは、アラートダイアログ/確認ダイアログ/入力ダイアログを表示する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
  </body>
</html>
```

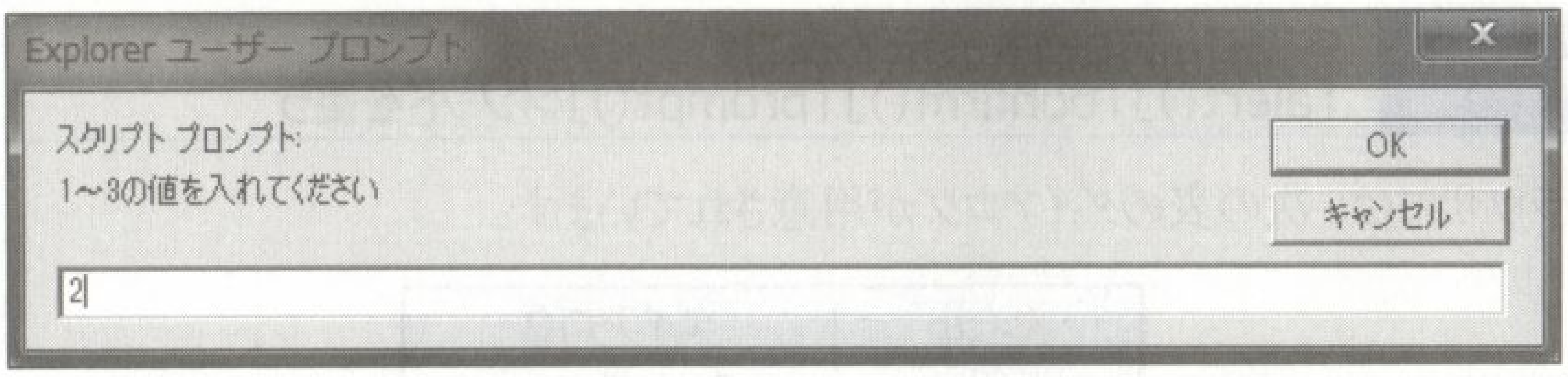
SAMPLE CODE JavaScriptのコード(sample.js)

```
  // 1～3の数値をあてる(数あてゲーム)
  var num = Math.floor(Math.random() * 3) + 1;
  // 入力ダイアログを表示
  var text = prompt("1～3の値を入れてください", 2);
  // 入力されていない場合はnullになるので以後の処理をしない
  if (!text){ return; }
  // 確認ダイアログを表示
  var flag = confirm("本当にこの数値でいいですか？");
  if (flag === false){ return; }
  // 当たっているか調べる(型は自動変換される)
  if (num == text){
    alert("正解です");
  }else{
    alert("不正解です。正解は"+num+"でした");
  }
}, false);
```

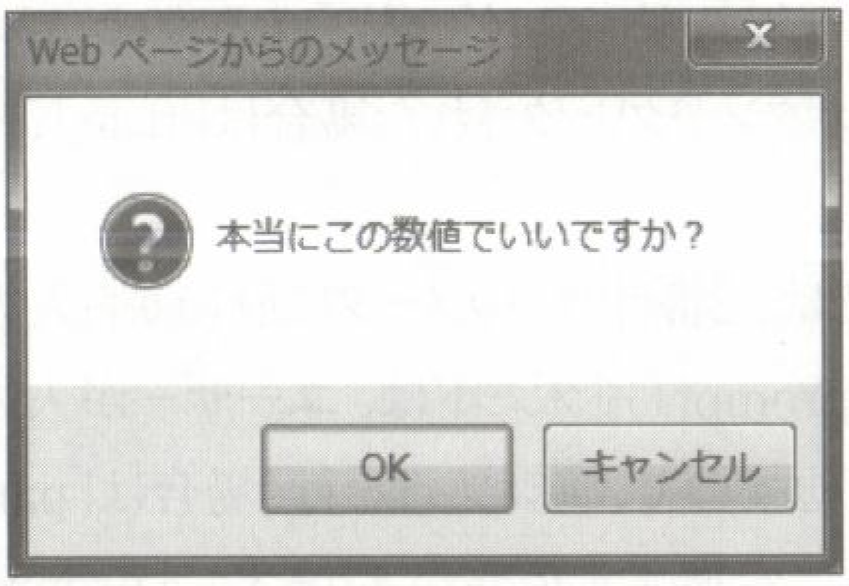
HINT

Windows Phoneでは「prompt()」メソッドを使うことができません(280ページのCOLUMNを参照)。

ブラウザでindex.htmlを表示すると、入力ダイアログが表示されるので1～3までの数値を入力し、「OK」ボタンをクリックします。「キャンセル」ボタンをクリックすると、以後の処理を行われません。



「OK」ボタンがクリックすると、確認のダイアログが表示されます。「キャンセル」ボタンをクリックすると、以後の処理を行われません。



正解の場合には、正解を示すアラートダイアログが表示されます。



不正解の場合は、正解の値がアラートダイアログに表示されます。



ONEPOINT

各種ダイアログを表示するには「alert()」「confirm()」「prompt()」メソッドを使う

ブラウザには、次の表のダイアログが用意されています。

メソッド	ダイアログ
alert()	アラートダイアログ
confirm()	確認ダイアログ
prompt()	入力ダイアログ

いずれも最初のパラメータは、ダイアログに表示するメッセージを指定します。「confirm()」メソッドでは、「はい」ボタンがクリックされた場合は「true」、そうでない場合は「false」を返します。

「prompt()」メソッドでは、2番目のパラメータに最初から入力済みの文字列を指定しておくことができます。「prompt()」メソッドは、ユーザーが入力した文字列を返します。数値であっても文字型で返すので、演算処理を行う場合は「parseInt()」や「parseFloat()」などを使って数値型に変換する必要があります。「prompt()」で「キャンセル」ボタンがクリックされた場合には、戻り値は「null」になります。そのため、文字列が入力されたかどうかは、「prompt()」の戻り値が「null」かどうかを調べればよいことになります。

COLUMN

表示できない文字

文字によっては、ダイアログに表示できない文字があります。書体に登録されていない場合や、サロゲートペアを利用した文字はダイアログに表示されません。なお、サロゲートペアを利用している文字を表示する場合は、195ページを参照してください。

COLUMN

Windows Phoneでは「prompt()」メソッドは使えない

Windows Phone(7.5)に搭載されているブラウザ(IE9)では「alert()」メソッドと「confirm()」メソッドは使えますが、「prompt()」メソッドは使うことができません。「prompt()」メソッドを呼び出しても画面には何も表示されず、エラーメッセージなども出ません。

関連項目 ▶▶▶

● モーダルダイアログを表示する ……………………………… p.281

SECTION-118

モーダルダイアログを表示する

ここでは、モーダルダイアログを表示する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

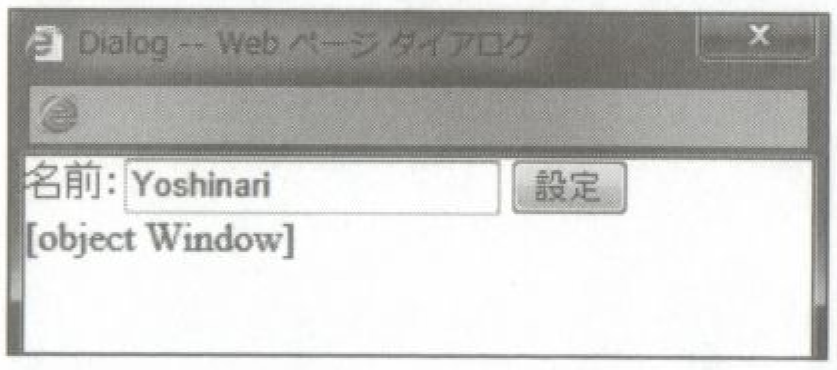
SAMPLE CODE HTMLのコード(input.html)

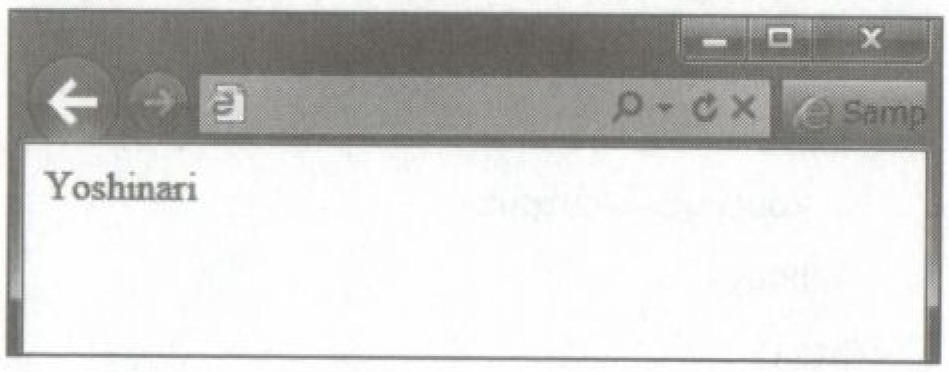
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Dialog</title>
</head>
<body>
<form>
名前：<input type="text" name="uName" id="uName">
<input type="button" value="設定" id="setBtn">
</form>
<output id="result"></output>
<script>
  // ボタンにイベントを設定
  var btn = document.getElementById("setBtn");
  btn.addEventListener("click", function(){
    var text = document.getElementById("uName").value  // 入力された内容を返す
    winObject.returnValue = text;  // 呼び出し元に返すデータ
    window.close();  // ダイアログを閉じる
  }, false);
  // 呼び出し元から渡されたパラメータを表示
  var winObject = window.dialogArguments;
  document.getElementById("result").innerHTML = winObject;
</script>
</body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // モーダルダイアログを表示
  var text = showModalDialog("./input.html", window,"dialogWidth=320px; dialogHeight=240px;");
  // 入力結果を表示
  ele.innerHTML = window.returnValue;
}, false);
```

ブラウザでindex.htmlを表示するとモーダルダイアログが表示され、名前を入力して「設定」ボタンをクリックすると、モーダルダイアログの呼び出し元にデータが返され、モーダルダイアログから返された文字列がページ上に表示されます。





ONEPOINT モーダルダイアログを表示するには「showModalDialog()」メソッドを使う

モーダルダイアログを表示するには、「showModalDialog()」メソッドを使います。「showModalDialog()」メソッドは、呼び出したダイアログでの処理が終わるまで、次の行に処理が移りません。

「showModalDialog()」メソッドの最初のパラメータには、ダイアログ内に表示するHTMLファイルのURLを指定します。2番目のパラメータには、モーダルダイアログに渡すパラメータを指定します。このとき、「window」オブジェクトを指定することで呼び出し元とデータのやり取りを行うこともできます。2番目のオプションは省略することもできます。

3番目のパラメータは、次の表に示すオプションを指定することができます。これも省略することができます。

オプション	説明
center	ダイアログを中央に表示するかどうか(yes/no)
dialogWidth	ダイアログの横幅
dialogheight	ダイアログの縦幅
dialogLeft	ダイアログのX座標(画面全体での位置)
dialogTop	ダイアログのY座標(画面全体での位置)
resizabel	リサイズ可能かどうか(yes/no)
scroll	スクロールバーを表示するかどうか(yes/no)
edge	まわりのエッジ(枠)を表示するかどうか(yes/no)。IEのみ
status	ステータスバーを表示するかどうか(yes/no)。IEのみ
unadorned	外観指定(yes/no)。IEのみ

関連項目 ▶▶▶

● アラートダイアログ/確認ダイアログ/入力ダイアログを表示する ………………… p.278

SECTION-119

タイマーを設定・解除する

ここでは、タイマーを設定・解除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

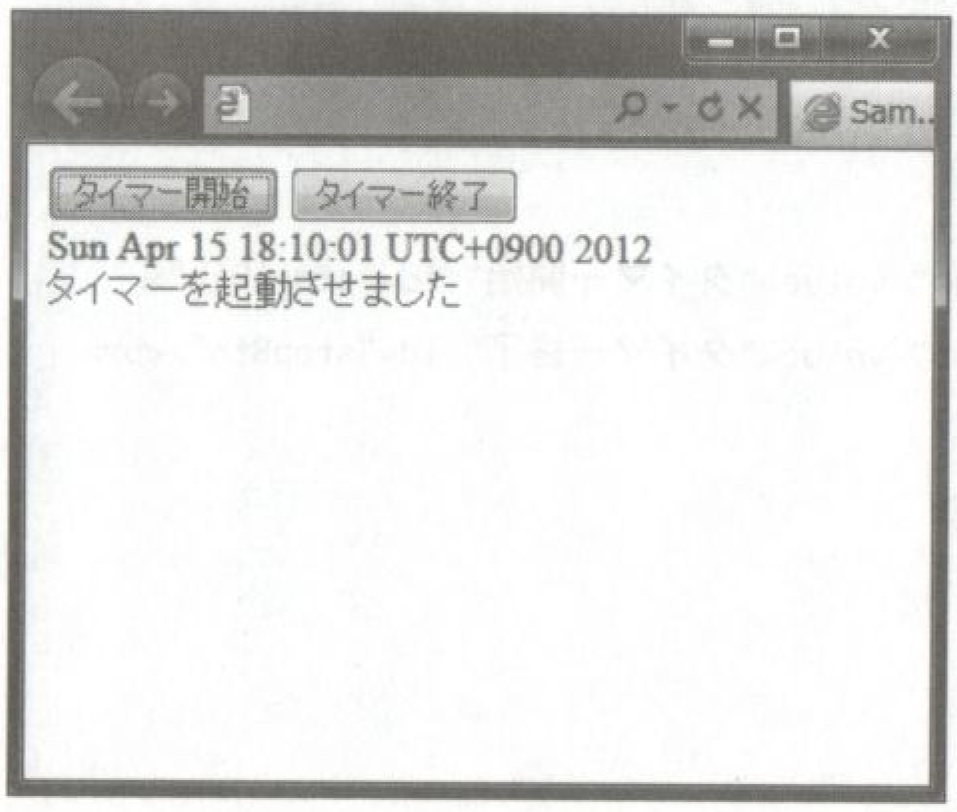
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="タイマー開始" id="startBtn">
      <input type="button" value="タイマー終了" id="stopBtn"><br>
    </form>
    <output></output><br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

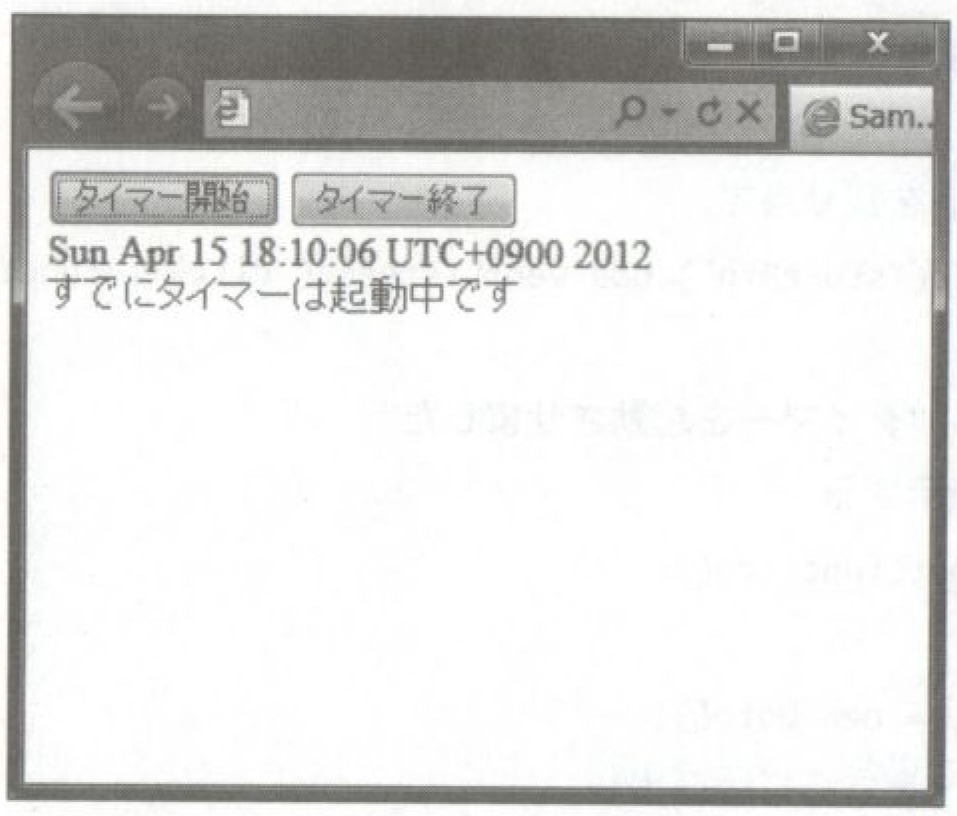
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // タイマーIDを入れる変数を用意
  var timerID = null;
  // 開始ボタンにイベントを割り当て
  document.getElementById("startBtn").addEventListener("click", function(){
    if (!timerID){
      ele[1].innerHTML = "タイマーを起動させました";
      // 1秒ごと時間を表示する
      timerID = setTimeout(function(){
        // 時間を表示
        ele[0].innerHTML = new Date();
        // 現在のタイマーをさらに呼び出し
        timerID = setTimeout(arguments.callee, 1000);
      }, 1000);
    }else{
      ele[1].innerHTML = "すでにタイマーは起動中です";
    }
  }, false);
  // 終了ボタンにイベントを割り当て
  document.getElementById("stopBtn").addEventListener("click", function(){
```

```
    if (!timerID){
      ele[1].innerHTML = "タイマーは停止しています";
    }else{
      ele[1].innerHTML = "タイマーを停止させました";
      clearTimeout(timerID);
      timerID = null;
    }
  }, false);
}, false);
```

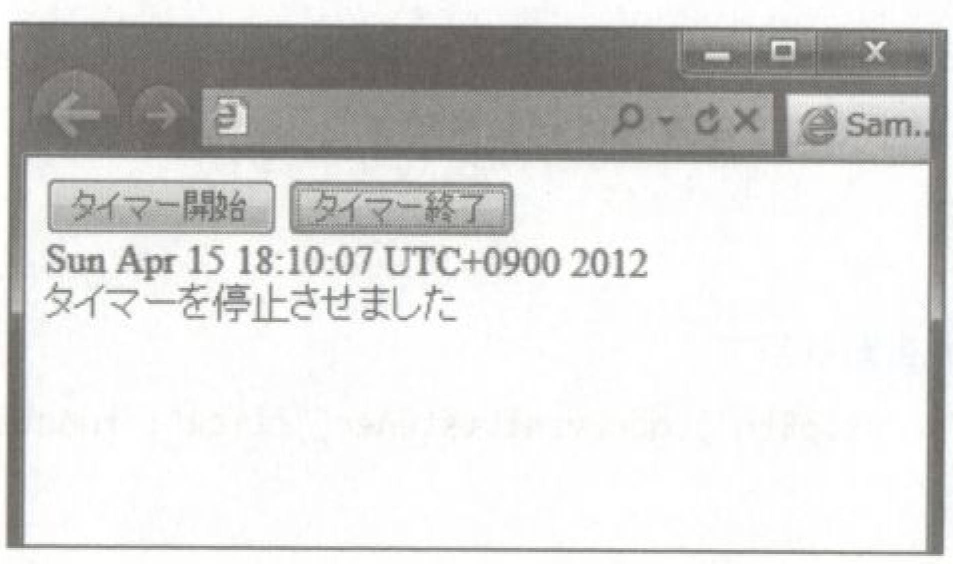
ブラウザでindex.htmlを表示し、「タイマー開始」ボタンをクリックすると、タイマーがスタートして現在時刻が1秒ごとに表示されます。



さらに「タイマー開始」ボタンをクリックすると、タイマーが動作中であるメッセージが表示されます。



「タイマー終了」ボタンをクリックすると、タイマーが停止します。



ONEPOINT

タイマーを設定するには「setTimeout()」メソッド、タイマーを解除するには「clearTimeout()」メソッドを使う

指定時間後に処理を行うには、「setTimeout()」メソッドを使います。「setTimeout()」メソッドの最初のパラメータに実行する処理(文字列で命令を囲むか、無名関数で処理を記述する)、2番目のパラメータにはミリ秒を指定します。ただし、2番目に指定したミリ秒後に必ずしも処理が実行される保証はありません。環境によっては、指定したミリ秒より後に実行されることがあります(指定したミリ秒より前に実行されることはない)。「setTimeout()」でタイマーを設定すると、戻り値としてタイマーIDが返されます。

タイマーを停止させる場合は「clearTimeout()」メソッドを使い、パラメータにタイマーIDを指定します。このタイマーIDは、「setTimeout()」メソッドの戻り値になります。

COLUMN

「setTimeout()」メソッドの3番目以降のパラメータ

「setTimeout()」メソッドには、3番目以降のパラメータを指定することができます(IEを除く)。3番目以降のパラメータは、「setTimeout()」メソッドで呼び出される関数のパラメータとして渡されます。たとえば、次のようにすると、1秒後にアラートダイアログにメッセージが表示されます。

```
timerID = setTimeout(function(msg){
  alert(msg);
}, 1000, "test message");
```

関連項目 ▶▶▶

- インターバルタイマーを設定・解除する ………… p.286
- アニメーション同期タイマーを設定・解除する ………… p.289
- タブを切り換えた場合に処理を行う ………… p.859

SECTION-120

インターバルタイマーを設定・解除する

ここでは、インターバルタイマーを設定・解除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

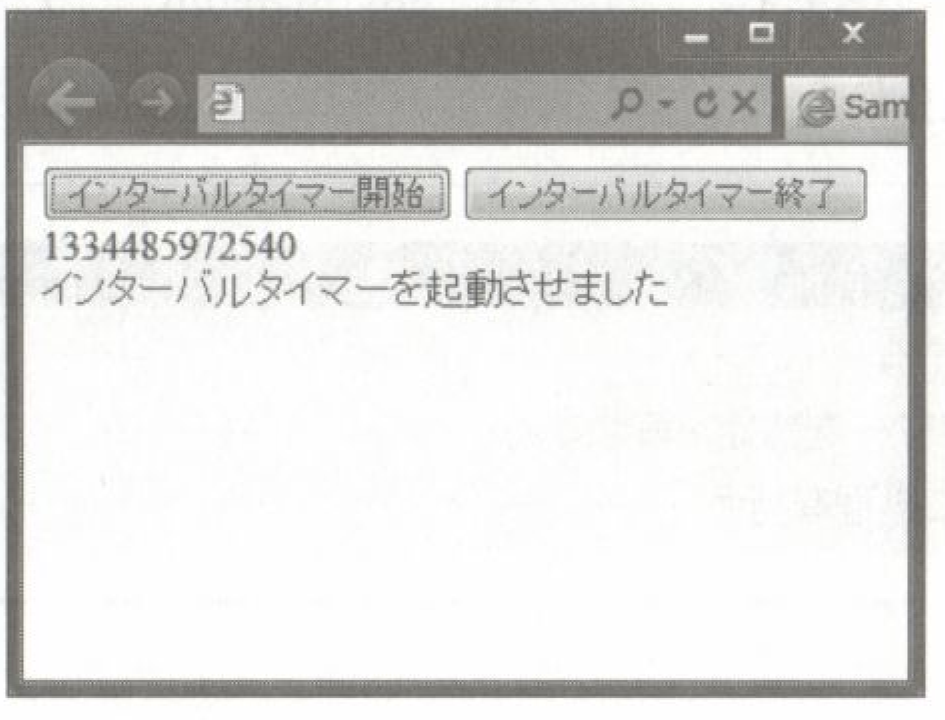
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="インターバルタイマー開始" id="startBtn">
      <input type="button" value="インターバルタイマー終了" id="stopBtn"><br>
    </form>
    <output></output><br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

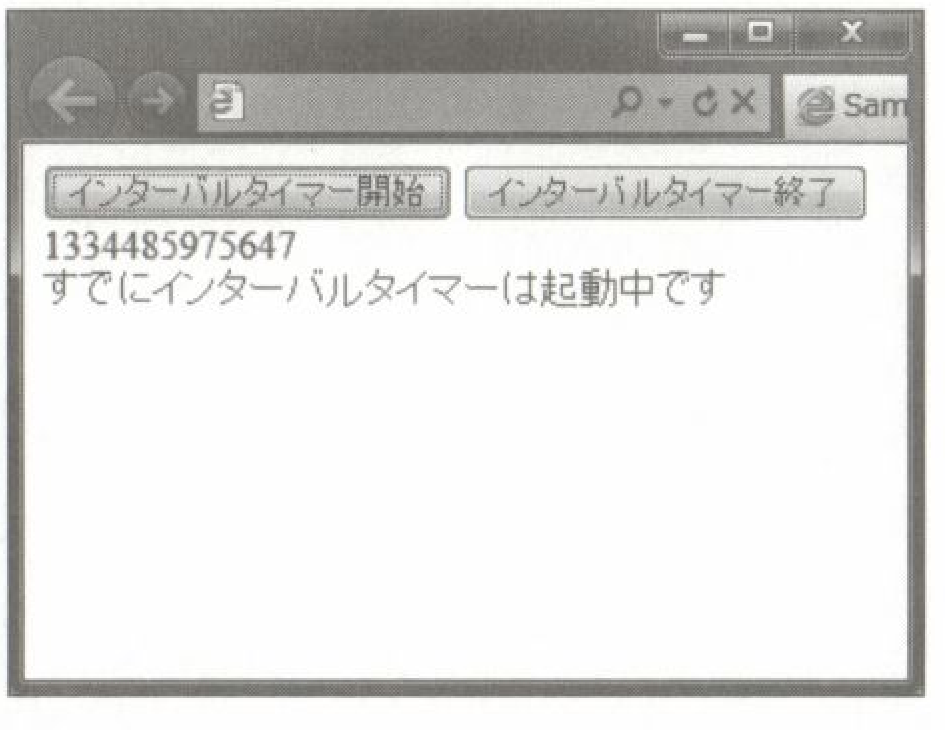
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // タイマーIDを入れる変数を用意
  var timerID = null;
  // 開始ボタンにイベントを割り当て
  document.getElementById("startBtn").addEventListener("click", function(){
    if (!timerID){
      ele[1].innerHTML = "インターバルタイマーを起動させました";
      // 1秒ごと定期的にミリ秒を表示する
      timerID = setInterval("displayTime()", 100);
    }else{
      ele[1].innerHTML = "すでにインターバルタイマーは起動中です";
    }
  }, false);
  // 終了ボタンにイベントを割り当て
  document.getElementById("stopBtn").addEventListener("click", function(){
    if (!timerID){
      ele[1].innerHTML = "インターバルタイマーは停止しています";
    }else{
      ele[1].innerHTML = "インターバルタイマーを停止させました";
      clearInterval(timerID);
```

```
      timerID = null;
    }
  }, false);
}, false);
// 定期的に呼び出される処理
function displayTime(){
  var ele = document.getElementsByTagName("output");
  ele[0].innerHTML = (new Date()).getTime();
}
```

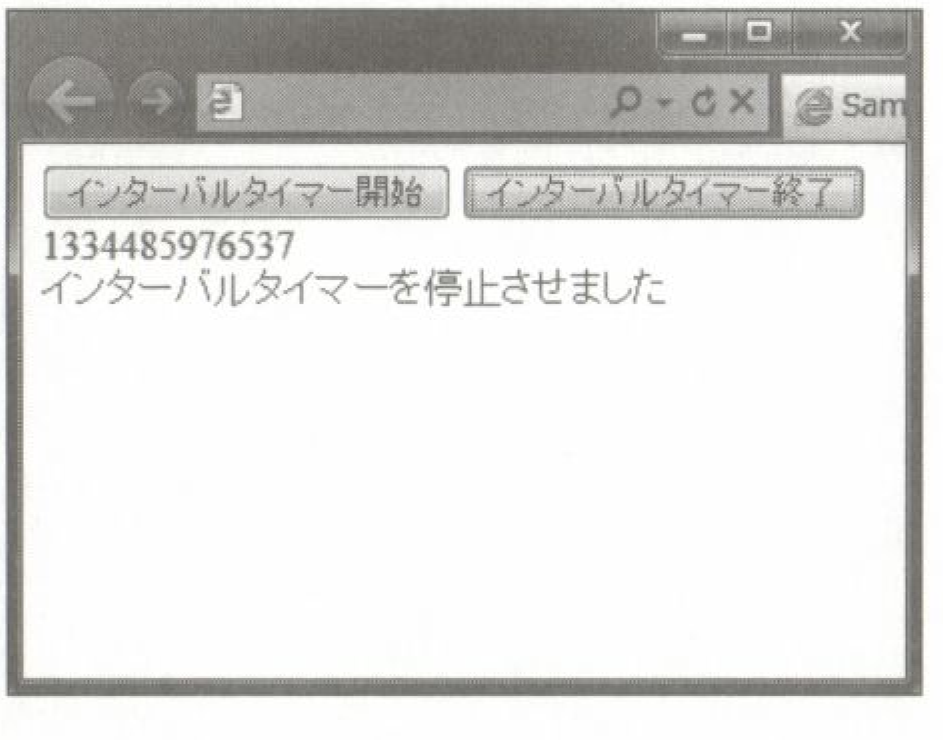
ブラウザでindex.htmlを表示し、「インターバルタイマー開始」ボタンをクリックすると、インターバルタイマーがスタートして現在のミリ秒が表示されます。



さらに「インターバルタイマー開始」ボタンをクリックすると、インターバルタイマーが動作中であるメッセージが表示されます。



「インターバルタイマー終了」ボタンをクリックすると、インターバルタイマーが停止します。



ONEPOINT インターバルタイマーを設定するには「setInterval()」メソッド、解除するには「clearInterval()」メソッドを使う

一定時間ごとに処理を行うには、「setInterval()」メソッドを使います。「setInterval()」メソッドの最初のパラメータに実行する処理(文字列で命令を囲むか、無名関数で処理を記述する)、2番目のパラメータには定期的に処理するミリ秒を指定します。ただし、2番目に指定したミリ秒ごとに必ずしも処理が実行される保証はありません。「setInterval()」でタイマーを設定すると、戻り値としてタイマーIDが返されます。

タイマーを停止させる場合は「clearInterval()」メソッドを使い、パラメータにタイマーIDを指定します。このタイマーIDは、「setInterval()」メソッドの戻り値になります。

なお、「setInterval()」メソッドも「setTimeout()」メソッドと同様に、3番目以降のパラメータを指定することができます。この場合、「setInterval()」メソッドで呼び出される関数にパラメータとして渡されます。

関連項目 ▶▶▶

- タイマーを設定・解除する ……… p.283
- アニメーション同期タイマーを設定・解除する ……… p.289
- タブを切り換えた場合に処理を行う ……… p.859

IE 10 Chrome Safari 5 Firefox 4 Opera 11 iOS 4 Android 2 WP

SECTION-121

アニメーション同期タイマーを設定・解除する

ここでは、アニメーション同期タイマーを設定・解除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

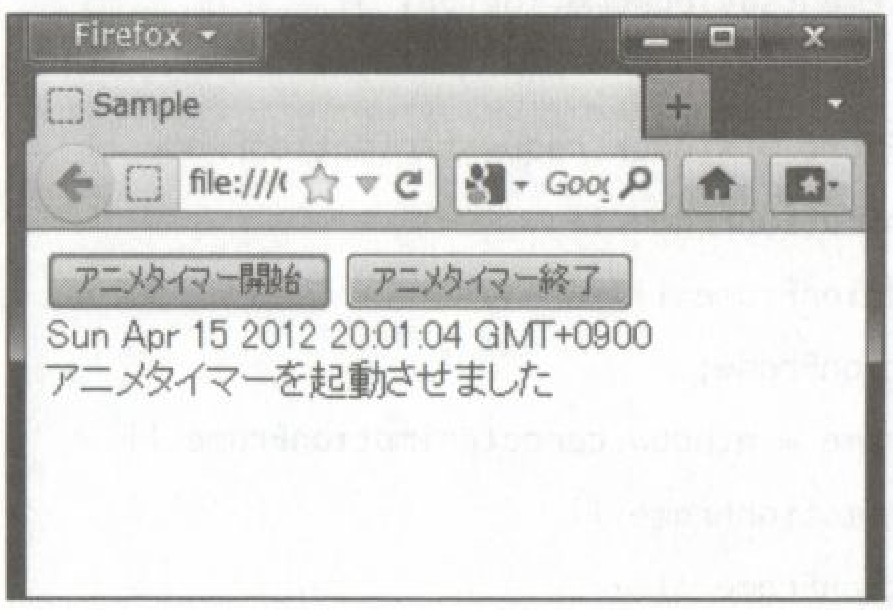
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="アニメタイマー開始" id="startBtn">
      <input type="button" value="アニメタイマー終了" id="stopBtn"><br>
    </form>
    <output></output><br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

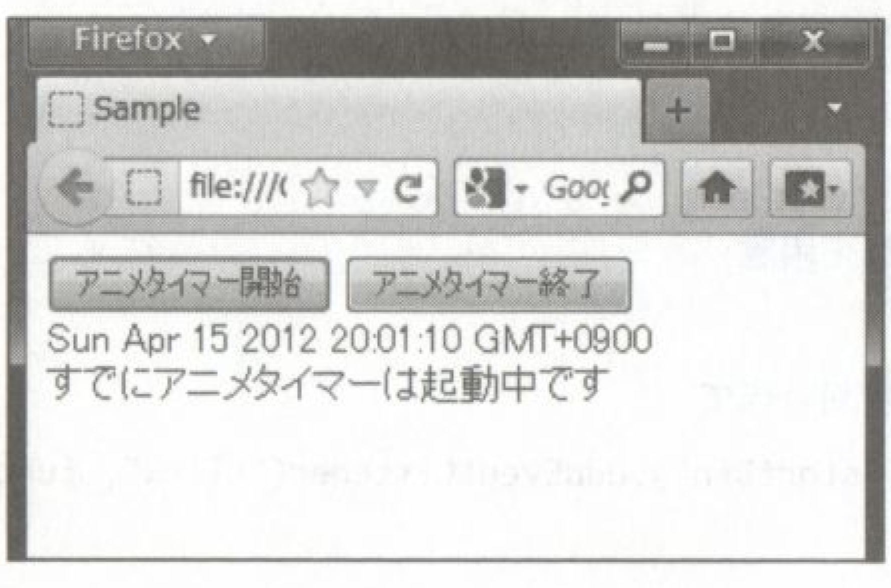
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // ベンダープレフィックスを考慮
  window.requestAnimationFrame = window.requestAnimationFrame ||
    window.webkitRequestAnimationFrame ||
    window.mozRequestAnimationFrame ||
    window.msRequestAnimationFrame;
  window.cancelAnimationFrame = window.cancelAnimationFrame ||
    window.webkitRequestAnimationFrame ||
    window.mozCancelAnimationFrame ||
    window.msRequestAnimationFrame;
  // チェック
  if (!window.requestAnimationFrame){
    ele[0].innerHTML = "機能が実装されていません";
    return;
  }
  // タイマーIDを入れる変数を用意
  var timerID = null;
  // 開始ボタンにイベントを割り当て
  document.getElementById("startBtn").addEventListener("click", function(){
    if (!timerID){
```

```
      ele[1].innerHTML = "アニメタイマーを起動させました";
      // 1秒ごと時間を表示する
      timerID = window.requestAnimationFrame(function(){
        // 時間を表示
        ele[0].innerHTML = new Date();
        // 現在のタイマーをさらに呼び出し
        timerID = window.requestAnimationFrame(arguments.callee);
      });
    }else{
      ele[1].innerHTML = "すでにアニメタイマーは起動中です";
    }
  }, false);
  // 終了ボタンにイベントを割り当て
  document.getElementById("stopBtn").addEventListener("click", function(){
    if (!timerID){
      ele[1].innerHTML = "アニメタイマーは停止しています";
    }else{
      ele[1].innerHTML = "アニメタイマーを停止させました";
      window.cancelAnimationFrame(timerID);
      timerID = null;
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「アニメタイマー開始」ボタンをクリックすると、タイマーがスタートして現在日時が1秒ごとに表示されます。



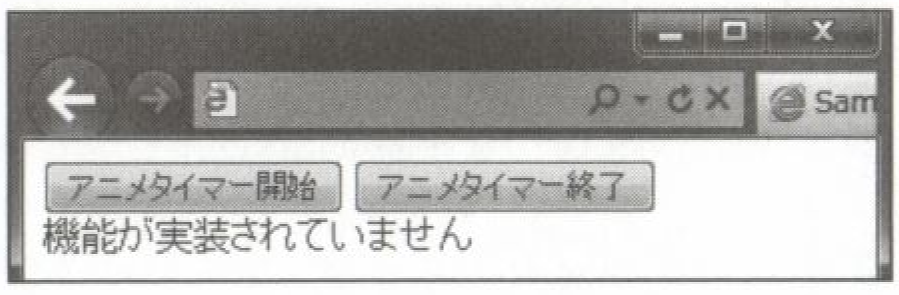
さらに「アニメタイマー開始」ボタンをクリックすると、タイマーが動作中であるメッセージが表示されます。



「アニメタイマー終了」ボタンをクリックすると、タイマーが停止されます。



実装されていない場合は、次のようなメッセージが表示されます。



ONEPOINT アニメーション同期タイマーを設定するには「window.requestAnimationFrame()」メソッド、解除するには「window.cancelAnimationFrame()」メソッドを使う

「setTimeout()」メソッドや「setInterval()」メソッドで個別にアニメーションさせると、それぞれ処理されてしまうため、描画タイミングが合わなかったり、負荷の原因になったりする場合があります。そこで、描画タイミングに合わせて処理するタイマーが「window.requestAnimationFrame()」メソッドです。

「window.requestAnimationFrame()」メソッドのパラメータに実行する処理（文字列で命令を囲むか、無名関数で処理を記述する）を指定します。「window.requestAnimationFrame()」メソッドは戻り値としてタイマーIDを返します。

「window.requestAnimationFrame()」メソッドで実行したタイマーを停止させるには、「window.cancelAnimationFrame()」メソッドを使います。パラメータにはタイマーIDを指定します。このタイマーIDは、「window.requestAnimationFrame()」メソッドの戻り値になります。なお、「window.mozCancelAnimationFrame()」メソッドは、Firefox 11以降で対応しています。

COLUMN ベンダープレフィックスについて

「requestAnimationFrame()」メソッドに限らず、試験的な機能に関しては、ベンダープレフィックスがメソッドやプロパティ名の先頭に付く場合があります。どのようなプレフィックスが付くかは、次のように決まっています。

ベンダープレフィックス	ブラウザ
webkit	WebKit。SafariやGoogle Chrome
moz	Firefox
ms	IE

関連項目 ▶▶▶

● タイマーを設定・解除する ………… p.283
● インターバルタイマーを設定・解除する ………… p.286
● タブを切り換えた場合に処理を行う ………… p.859

SECTION-122

非同期処理を行う

ここでは、「setImmediate()」メソッドを使って非同期処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

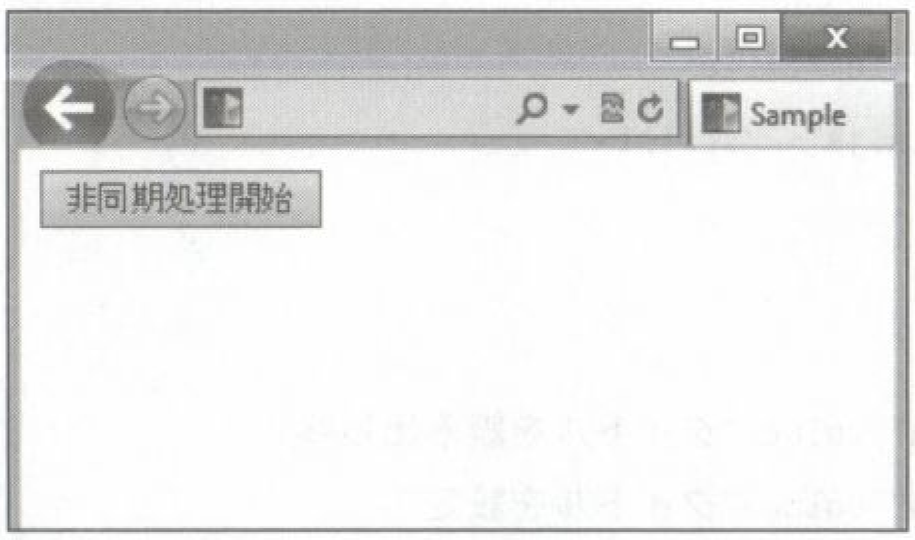
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="非同期処理開始" id="startBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ベンダープレフィックスを考慮
  window.setImmediate = window.setImmediate || window.msSetImmediate;
  window.clearImmediate = window.clearImmediate || window.msClearImmediate;
  // 開始ボタンにイベントを割り当て
  document.getElementById("startBtn").addEventListener("click", function(){
    // 非同期処理1を実行
    var immediateID = window.setImmediate(primeNumber);
    // 非同期処理2を実行
    window.setImmediate(primeNumber);
    // 非同期処理3を実行
    window.setImmediate(primeNumber);
    // 非同期処理4を実行
    window.setImmediate(primeNumber);
  }, false);
}, false);
// 時間がかかる処理
// 10000までの最大の素数を求める
function primeNumber() {
  var ele = document.getElementsByTagName("output")[0];
  var result = 2;
  for (var i=2; i<=10000; i++) {
    var flag = false;  // 素数フラグをリセット
```

```
    for (var j=2; j<i; j++) {
      if ((i % j) == 0) { flag = true; } // フラグをセット
    }
    if (flag == false) { result = i; }
  }
  ele.innerHTML += "素数計算が終了しました。"+result+" : "+Date.now()+"<br>";
}
```

ブラウザでindex.htmlを表示し、「非同期処理開始」ボタンをクリックすると、10000までの最大の素数を求める関数が非同期で4つ起動します。



非同期処理が完了すると、メッセージと最大の素数、そして時間（ミリ秒）が表示されます。



ONEPOINT　非同期処理を行うには「setImmediate()」メソッドを使う

JavaScriptで非同期処理を行うには、「setTimeout()」メソッドやWeb Workersを使う以外に、「setImmediate()」メソッドを使う方法があります。「setImmediate()」のパラメータに、非同期で処理したい関数名を指定します。「setImmediate()」メソッドを使い呼び出した処理を停止するには、「clearImmediate()」メソッドを使います。

COLUMN　「setTimeout()」メソッドを使った非同期処理

処理速度は低下しますが、「setTimeout(関数名, 0)」として非同期処理を行う方法もあります。また、Web Workersを使って非同期処理を行うこともできます。

関連項目 ▶▶▶

● ワーカーオブジェクトを生成する ………………………………………… p.751

SECTION-123

ドキュメントのタイトルを取得・設定する

ここでは、ドキュメントのタイトルを取得・設定する方法について解説します。

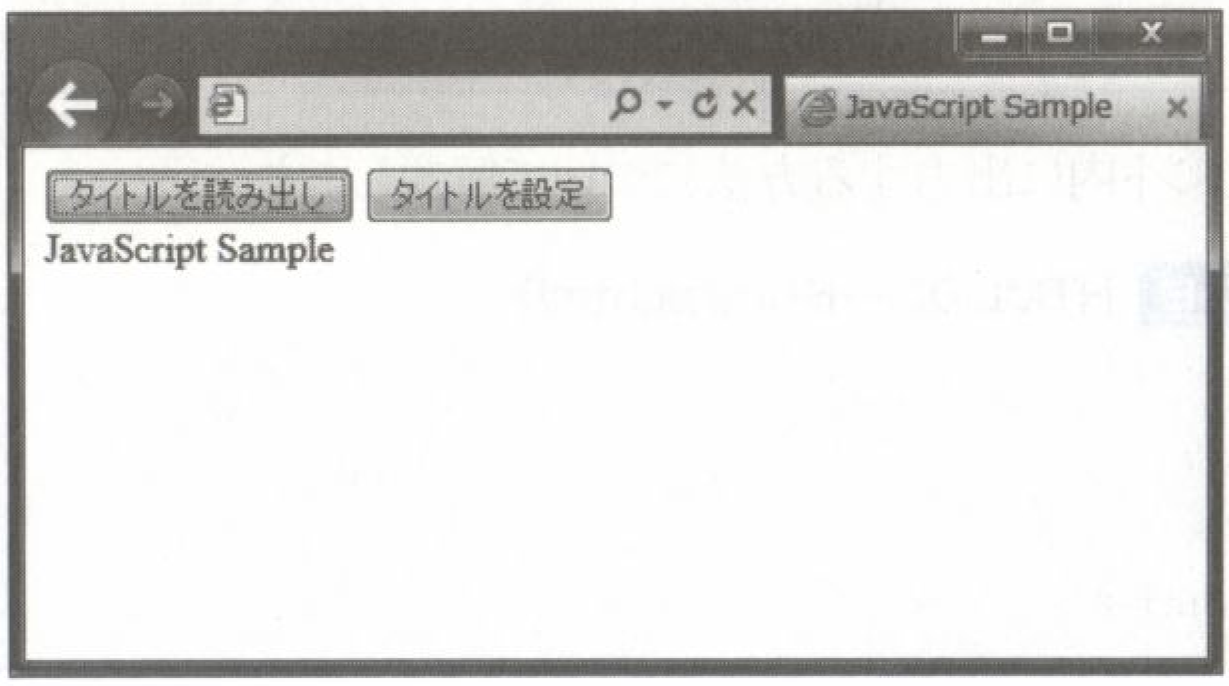
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>JavaScript Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="タイトルを読み出し">
      <input type="button" value="タイトルを設定">
    </form>
    <output></output>
  </body>
</html>
```

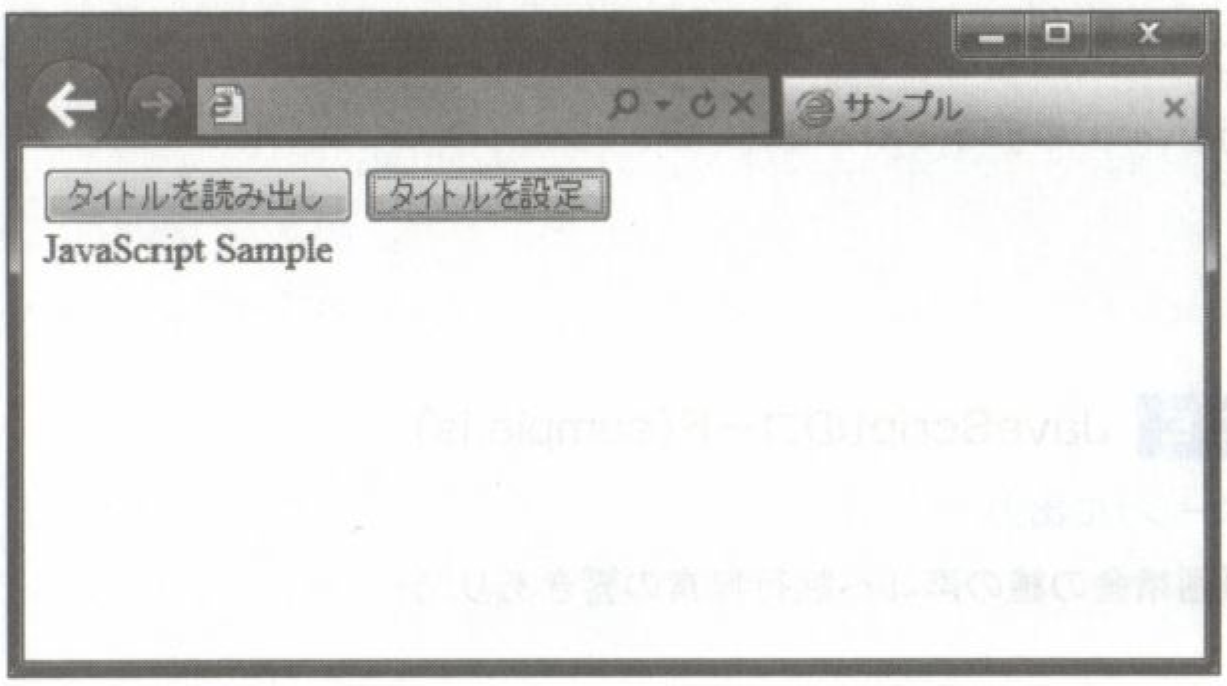
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // ドキュメントのタイトルを読み出し
    var text = document.title;
    // 結果を出力
    document.getElementsByTagName("output")[0].innerHTML = text;
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // ドキュメントのタイトルを設定
    document.title = "サンプル";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「タイトルを読み出し」ボタンをクリックすると、ドキュメントのタイトルが読み出されてページ上に表示されます。



「タイトルを設定」ボタンをクリックすると、「サンプル」の文字がタイトルとして設定されます。



ONEPOINT　ドキュメントのタイトルを取得・設定するには「title」プロパティを使う

ドキュメントのタイトルは「document」オブジェクトの「title」プロパティにアクセスすることで読み出し、設定することができます。なお、スマートフォン上で動作するブラウザではタイトルが表示されないものもあります。

IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-124

ドキュメント内に出力する

ここでは、ドキュメント内に出力する方法について解説します。

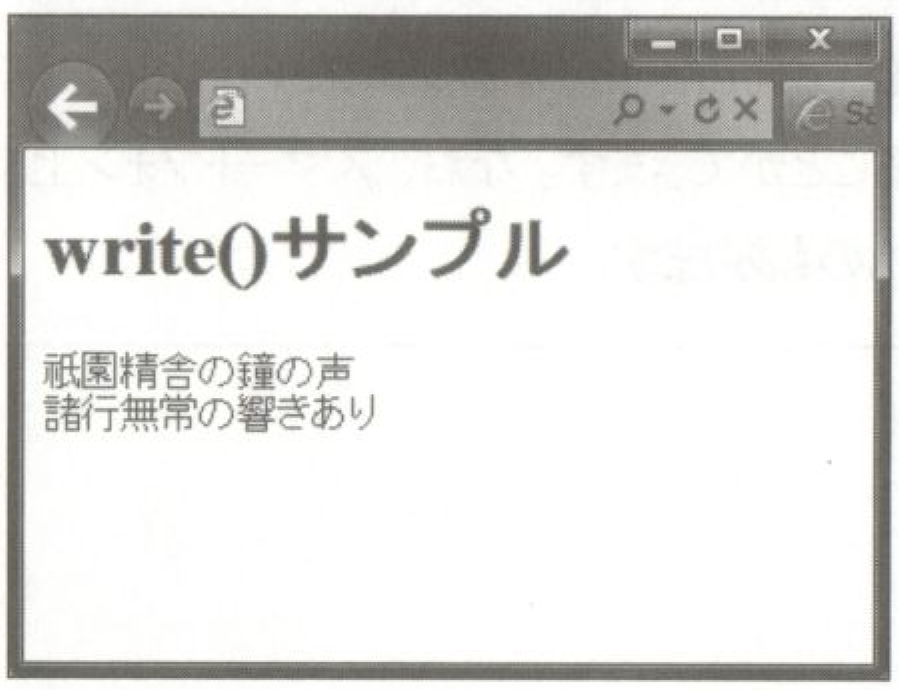
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
  </head>
  <body>
    <h1>write()サンプル</h1>
    <script src="js/sample.js"></script>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
// ドキュメント(ページ)に出力
document.write("祇園精舎の鐘の声<br>諸行無常の響きあり");
```

ブラウザでindex.htmlを表示すると、「document.write()」メソッドを使って出力した文字が表示されます。



ONEPOINT ページへの出力を行うには「document.write()」メソッドを使う

ページ上に文字や画像を出力する場合、大きく分けて2つの方法があります。1つは、ページ構築が完了する前に「document.write()」を使う方法です。もう1つは、ページ構築後にDOMを操作するメソッドを使う方法です(366ページ参照)。

「document.write()」は、ページ構築後に使用すると、予期せぬ結果を招くことがあります。また、「document.write()」を使っていない場合、「script」要素に「async」属性を指定することで、ページの構築処理を高速化することができます。

SECTION-125

クッキーに書き込みが可能かどうか調べる

ここでは、クッキーに書き込みが可能かどうか調べる方法について解説します。

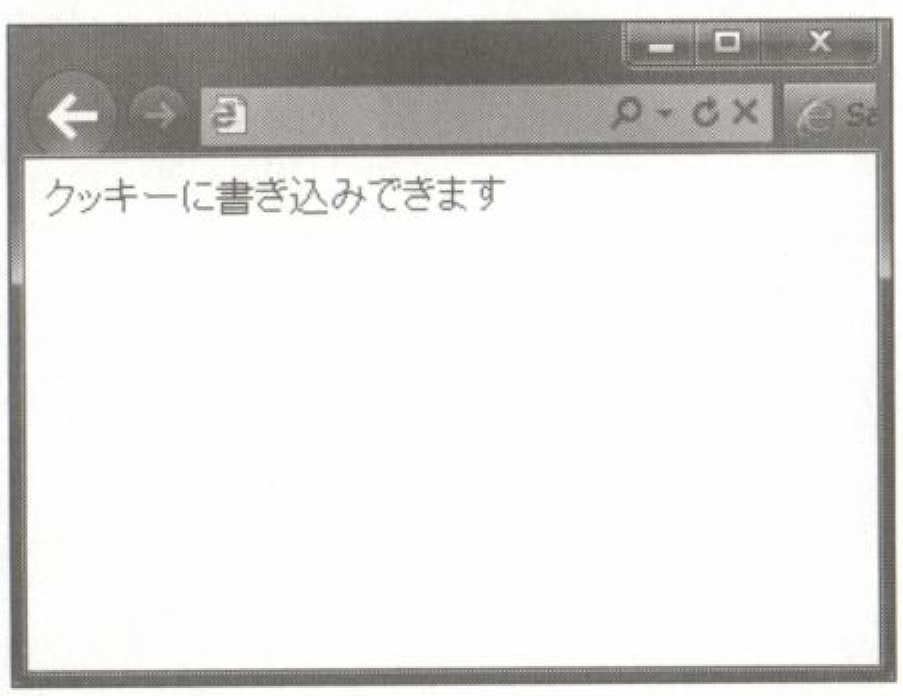
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // クッキーに書き込みができるかどうか調べる
  var flag = navigator.cookieEnabled;
  // 結果を出力
  if (flag === true){
    ele.innerHTML = "クッキーに書き込みできます";
  }else{
    ele.innerHTML = "クッキーに書き込めません";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、クッキーに書き込めるかどうかの結果が表示されます。



ONEPOINT クッキーが利用できるか調べるには「cookieEnabled」プロパティを使う

データを保持できるという点で、クッキーは古くから利用されています。このクッキーに書き込みができるかどうかは、「navigator」オブジェクトの「cookieEnabled」プロパティを参照します。「true」であれば書き込みが可能、「false」であれば書き込みはできないことになります。

データを保存する際、クッキーには制限があります。より多くのデータを保存するような場合は、ローカルストレージを利用するのがよいでしょう。また、一時的なデータの保存であれば、セッションストレージを利用する方法もあります。また、ファイルとして保存するような場合は、File APIを利用するのがよいでしょう。

関連項目 ▶▶▶

● クッキーの読み書きを行う ………………………………………………………… p.299

SECTION-126

クッキーの読み書きを行う

ここでは、クッキーのデータの読み込みと書き込み方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>クッキーの読み書き</h1>
    <form>
      書き込む内容：<input type="text" id="cData" value="cookie...."><br>
      <input type="button" id="cWrite" value="クッキーに書き込み">
      <input type="button" id="cRead" value="クッキーから読み込み">
    </form>
    <output></output>
  </body>
</html>
```

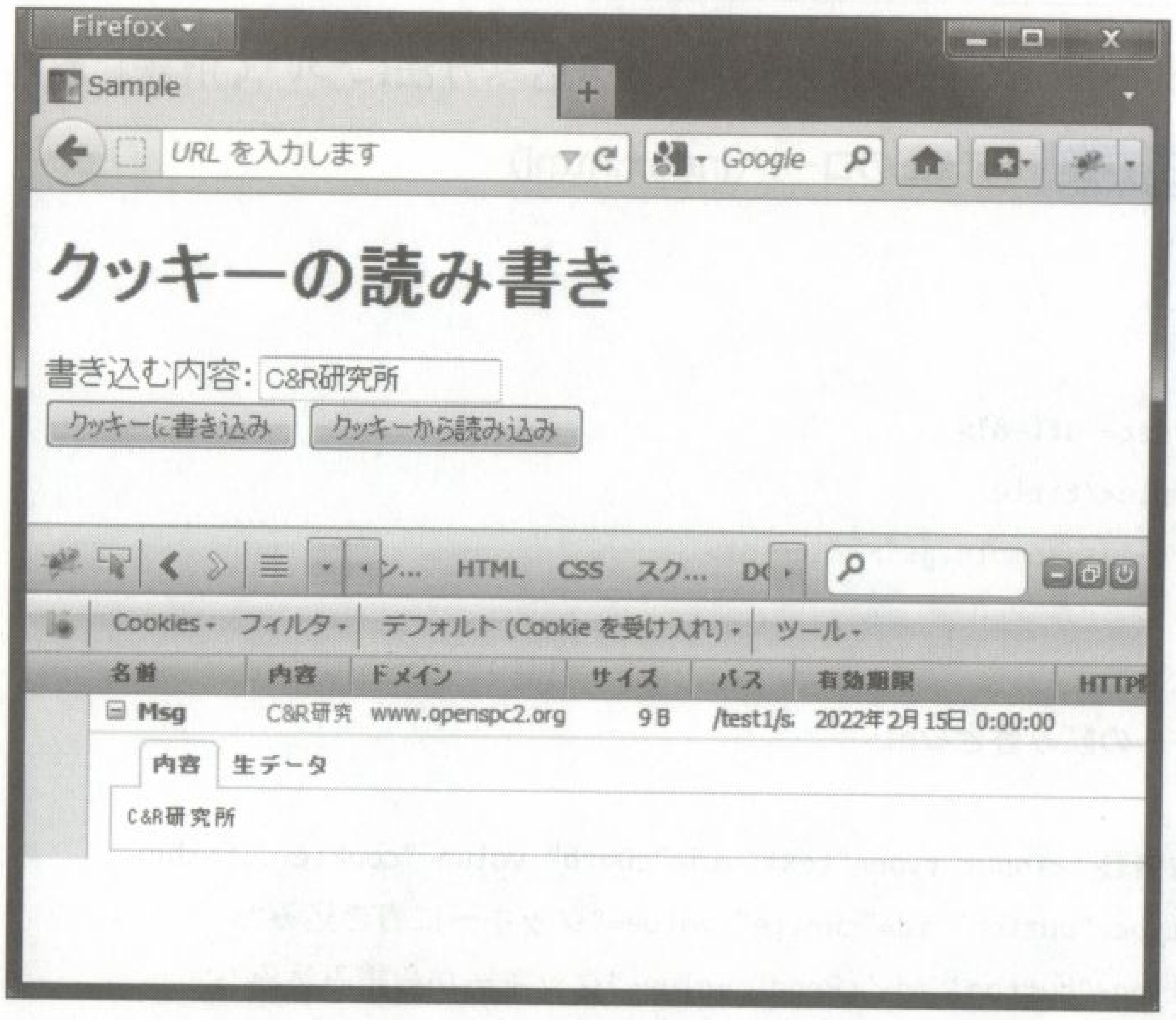
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 書き込みボタンにイベントを割り当て
  document.getElementById("cWrite").addEventListener("click", function(){
    var text = document.getElementById("cData").value;
    // クッキーに書き込み。期限を2022年2月15日に設定
    var expDay = (new Date("2022/2/15")).toGMTString();
    var data = "Msg="+escape(text)+";expires="+expDay;
    document.cookie = data;
  }, false);
  // 読み込みボタンにイベントを割り当て
  document.getElementById("cRead").addEventListener("click", function(){
    // クッキーから読み込み。変数dataに代入
    var data = document.cookie;
    ele.innerText = ele.textContent = unescape(data);
  }, false);
}, false);
```

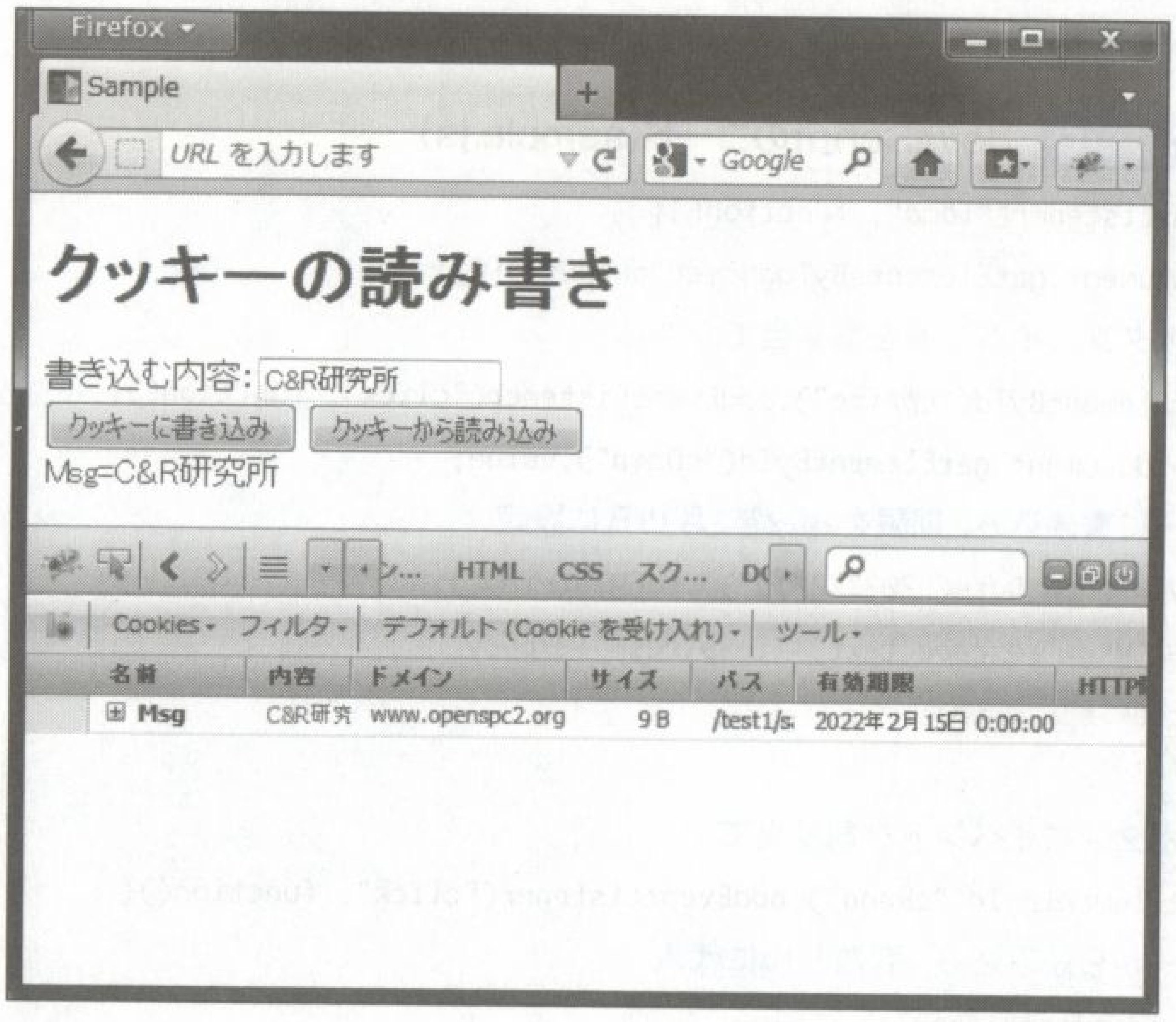
HINT

このサンプルは、ローカルディスク上では動作しないブラウザもあります。その場合は、ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、「書き込み」ボタンをクリックすると、クッキーにデータと期限が書き込まれます。これは、Firebug+Firecookieなどのデバッガで確認することができます（Firefox 13以降に対応したFirebugではFirecookieは不要）。



「読み込み」ボタンをクリックすると、クッキーに保存されているキーと値が読み出されて表示されます。



ONEPOINT クッキーの読み書きを行うには「document.cookie」プロパティを使う

クッキーの内容を読み出すには「document.cookie」プロパティにアクセスします。読み出される内容は「キー=内容」の文字列になり、同一ドメイン（もしくはパス）のデータになります。

クッキーに書き込むには、「キー名=内容;expires=期限」を「document.cookie」プロパティに書き込みます。複数のキーと内容を保存する場合は、「;」（セミコロン）で区切って列記します。また、保存するオプションは「expires」の期限だけでなく、次の表のオプションもあります。

オプション	説明
expires	期限。GMT（グリニッジ標準時/協定世界時）で指定
domain	ドメイン
path	パス
secure	セキュアな場合のみクッキーを送信

なお、クッキーへの書き込みはローカルディスク上では動作しない場合があります。

関連項目 ▶▶▶

● クッキーに書き込みが可能かどうか調べる ……………………………… p.297

SECTION-127

ドメイン名・ホスト名・パス名・プロトコルを求める

ここでは、ドメイン名・ホスト名・パス名・プロトコルを求める方法について解説します。

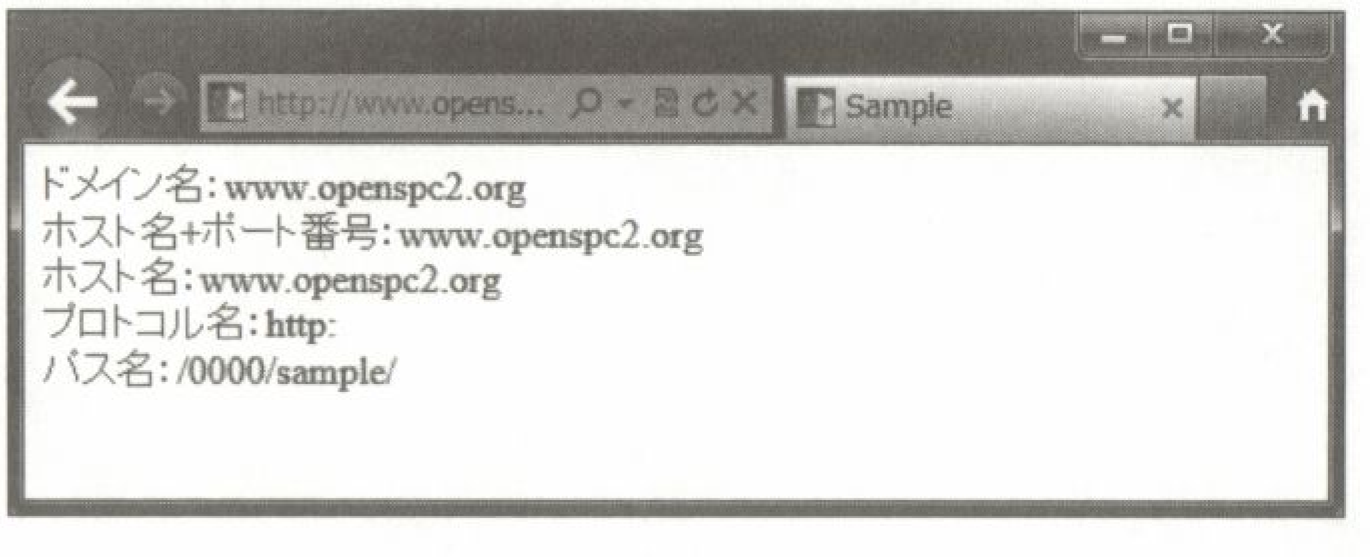
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ドメイン名を読み出し
  ele.innerHTML = "ドメイン名 :"+document.domain;
  // ホスト名+ポート番号を読み出し
  ele.innerHTML += "<br>ホスト名+ポート番号 :"+location.host;
  // ホスト名を読み出し
  ele.innerHTML += "<br>ホスト名 :"+location.hostname;
  // プロトコル名を読み出し
  ele.innerHTML += "<br>プロトコル名 :"+location.protocol;
  // パス名を読み出し
  ele.innerHTML += "<br>パス名 :"+location.pathname;
}, false);
```

ブラウザでindex.htmlを表示すると、現在のページのドメイン名・ホスト名・パス名・プロトコルが表示されます。



ONEPOINT

URL情報を取得するには「location」オブジェクトと「document」オブジェクトのプロパティを使う

現在、表示されているページのURLに関する情報（ドメイン名やプロトコル名など）は、「location」オブジェクトと「document」オブジェクトの各プロパティにあります。それぞれのプロパティと得られる情報は、表のようになります。

プロパティ	情報
document.domain	ドメイン名
location.host	ホスト名とポート番号
location.hostname	ホスト名
location.protocol	プロトコル名
location.pathname	ファイルのパス名

直前に見ていたページのURLを求める

ここでは、直前に見ていたページのURLを求める方法について解説します。

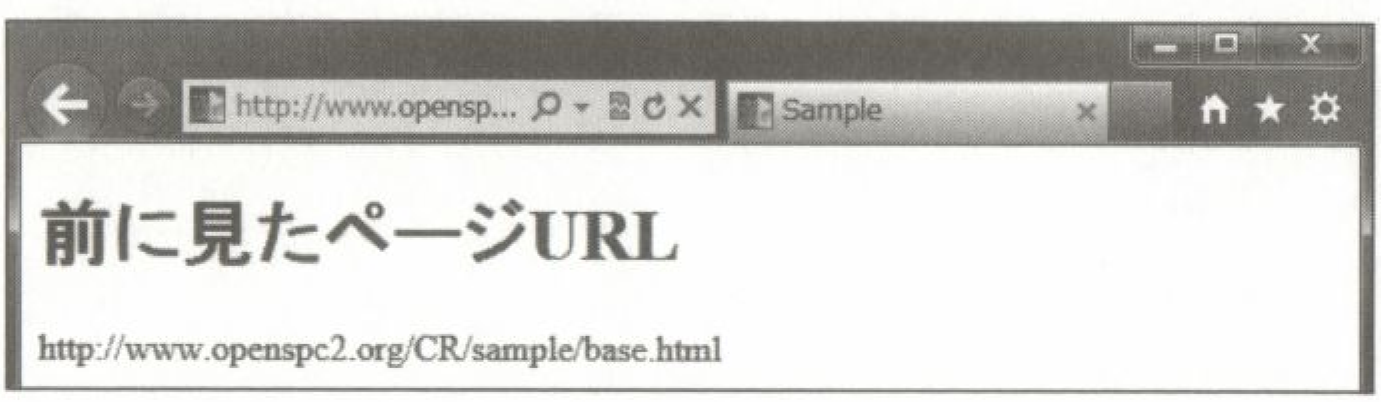
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>前に見たページURL</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 以前見ていたページURLを読み出す
  var url = document.referrer;
  // 結果を出力
  ele.innerHTML = url;
}, false);
```

ブラウザでindex.htmlを表示すると、直前に見ていたURLが表示されます。なお、サンプルの中にindex.htmlへのリンクがあるbase.htmlを用意してあるので、Webサーバーのindex.htmlと同じ階層にアップロードしてリンクをクリックすると、base.htmlのURLが表示されます。



ONEPOINT 前に見ていたページURLを取得するには「document.referrer」プロパティを使う

アクセス解析などを行う場合には、前に見ていたページのURLが重要になります。スクリプトでは、「document.referrer」プロパティを参照することで1つ前に見ていたページのURLを取得することができます。ただし、「document.referrer」プロパティは、前に見ていたページのURLを取得できない場合があります。

ページをリロードする

ここでは、ページをリロードする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ページリロード</h1>
    <form>
      <input type="button" value="リロード">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(){
    // trueを指定して強制的にサーバーから読み込み
    location.reload(true);
  }, false);
  // 現在の時間を出力
  document.getElementsByTagName("output")[0].innerHTML = new Date();
}, false);
```

ブラウザでindex.htmlを表示し、「リロード」ボタンをクリックすると、ページが再度、読み込まれます。



ONEPOINT ページをリロードするには「reload()」メソッドを使う

ページをリロードする場合には、「location」オブジェクトの「reload()」メソッドを使います。パラメータに「true」を指定した場合は、キャッシュではなく、サーバーからデータを読み込みます。パラメータを省略するか、「false」を指定した場合は、キャッシュがあれば、キャッシュを利用してページ内容が表示されます。

「location.reload()」メソッドがうまくいかない場合は、次のように、URLに現在の時間をURLパラメータとして指定する方法もあります。

```
location.href = "http://www.openspc2.org/index.html?t="+(new Date()).getTime();
```

関連項目 ▶▶▶

●タブを切り換えた場合に処理を行う ………………………………………… p.859

SECTION-130

ページのURLを取得・設定する

ここでは、ページのURLを取得・設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="URLを表示">
      <input type="button" value="指定URLに移動">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("input");
  // 最初のボタンにイベントを割り当て
  ele[0].addEventListener("click", function(){
    // 現在のページURLを取得
    var url = location.href;
    alert(url);
  }, false);
  // 2番目のボタンにイベントを割り当て
  ele[1].addEventListener("click", function(){
    location.href = "http://www.c-r.com/";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「URLを表示」ボタンをクリックすると、現在のページのURLが表示されます。



「指定URLに移動」ボタンをクリックすると、指定したURLのページに移動します。



ONEPOINT URLの取得と移動を行うには「href」プロパティを使う

現在のURL(URI)を取得するには、「location」オブジェクトの「href」プロパティを参照します。ページを移動したい場合には、「href」プロパティに移動先のURL(URI)を指定します。

SECTION-131

「Image」オブジェクトを生成する

ここでは、「Image」オブジェクトを生成する方法について解説します。

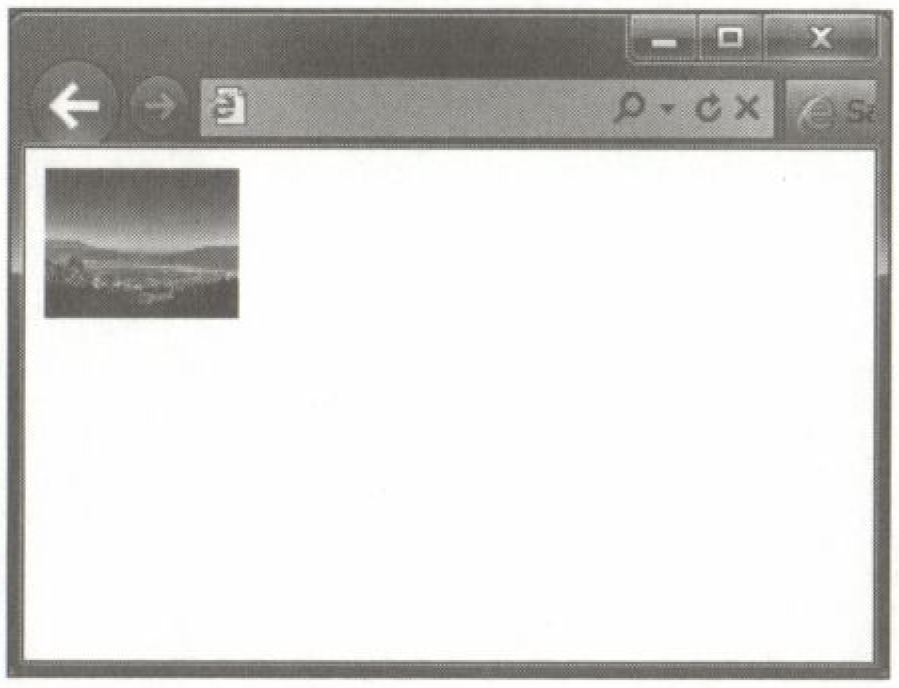
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 画像オブジェクトを生成
  var myImage = new Image(80, 60);
  // 画像のURLを設定する
  myImage.src = "images/photo1.jpg";
  // 結果を出力
  ele.appendChild(myImage);
}, false);
```

ブラウザでindex.htmlを表示すると、「Image」オブジェクトで生成された画像が表示されます。



ONEPOINT 「Image」オブジェクトを生成するには「new Image()」とする

画像(「Image」オブジェクト)を生成するには、「new Image()」とします。パラメータには、生成する画像の横幅と縦幅を指定します(単位はピクセル)。省略した場合は、オリジナルの画像の幅が適用されます。画像の幅や高さは、生成された「Image」オブジェクトの「width」プロパティや「height」プロパティを使って後から設定することもできます。

また、表示する画像は「src」プロパティにURL(URI)を代入します。注意点としては、画像は非同期で読み込まれるので、Canvasなどで描画する場合には、「Image」オブジェクトの「onload」プロパティにイベントハンドラを設定し、イベントハンドラ内で画像を描画するようにします。

なお、「new Image()」として画像を生成する以外に、「document.createElement("img")」として生成する方法もあります。この場合でも、「Image」オブジェクト(「img」要素)が生成されます(366ページ参照)。

関連項目

- 画像データが読み込まれた場合に処理する …… p.315
- 画像データを入れ替える …… p.317
- カメラで撮影した画像を表示する …… p.673
- 画像をフルスクリーン表示にする …… p.799
- 画像や映像ファイルを読み込む …… p.687

SECTION-132

画像の幅を設定・取得する

ここでは、画像の幅を設定・取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

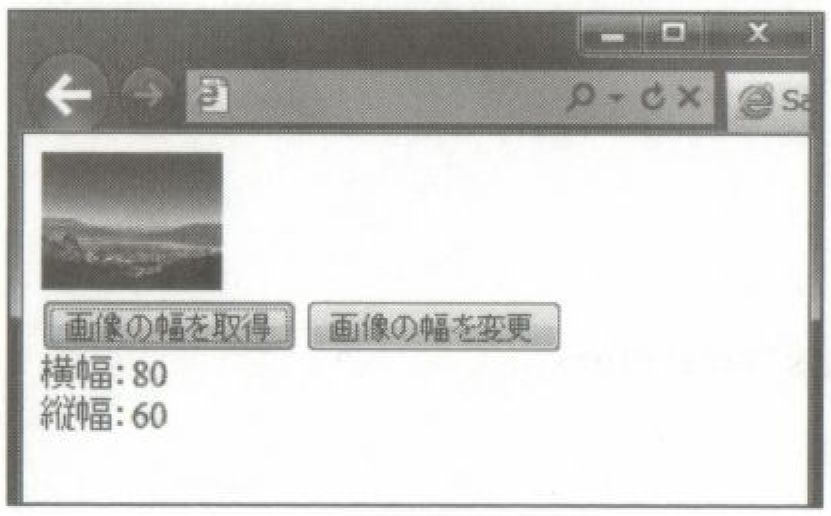
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <div>
      <img src="images/photo1.jpg" width="80" height="60">
    </div>
    <form>
      <input type="button" value="画像の幅を取得">
      <input type="button" value="画像の幅を変更">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

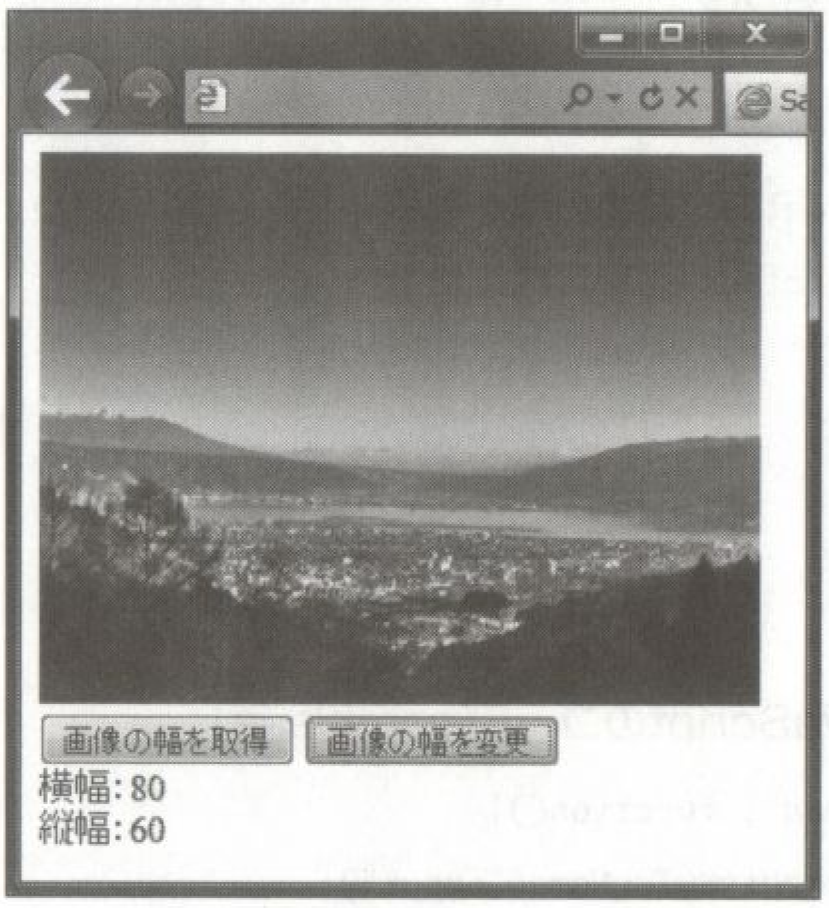
```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // img要素を読み出す
  var myImage = document.getElementsByTagName("img")[0];
  // ボタンにイベントを設定する
  btn[0].addEventListener("click", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 表示されている画像の横幅を読み出す
    var w = myImage.width;
    // 表示されている画像の縦幅を読み出す
    var h = myImage.height;
    // 結果を出力
    ele.innerHTML = "横幅 :"+w+"<br>縦幅 :"+h;
  }, false);
  // ボタンにイベントを設定する
  btn[1].addEventListener("click", function(){
    // 表示されている画像の横幅を320pxにする
    myImage.width = 320;
```

```
    // 表示されている画像の縦幅を240pxにする
    myImage.height = 240;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「画像の幅を取得」ボタンをクリックすると、画像の横幅と縦幅が表示されます。



「画像の幅を変更」ボタンをクリックすると、画像のサイズが変わります。



ONEPOINT **画像のサイズを取得・設定するには「width」プロパティと「height」プロパティを使う**

「Image」オブジェクト、または、ページ上に表示されている「img」要素のサイズは、「width」プロパティと「height」プロパティで取得と設定ができます。「width」プロパティは横幅、「height」プロパティは縦幅となり、「width」プロパティと「height」プロパティで取得・設定する値の単位はピクセルになります。「width」プロパティ、「height」プロパティだけでなく、CSSの「width」プロパティ、「height」プロパティを使って設定することもできます。その場合は、「"320pt"」のように単位付きで設定する必要があります。

また、画像のサイズは表示されているサイズになり、画像の元々のサイズを取得するには313ページで説明している「naturalWidth」プロパティや「naturalHeight」プロパティを利用します。

関連項目 ▶▶▶

● 画像のオリジナルの幅を取得する …… p.313

画像のオリジナルの幅を取得する

ここでは、画像のオリジナルの幅を取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

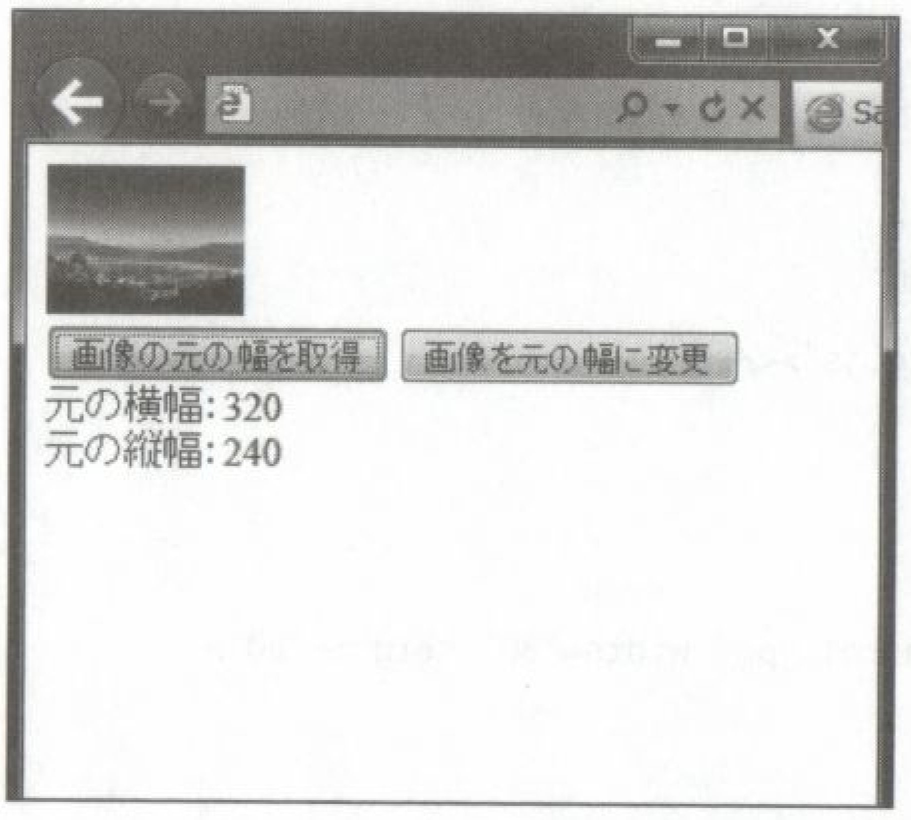
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <div>
      <img src="images/photo1.jpg" width="80" height="60">
    </div>
    <form>
      <input type="button" value="画像の元の幅を取得">
      <input type="button" value="画像を元の幅に変更">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

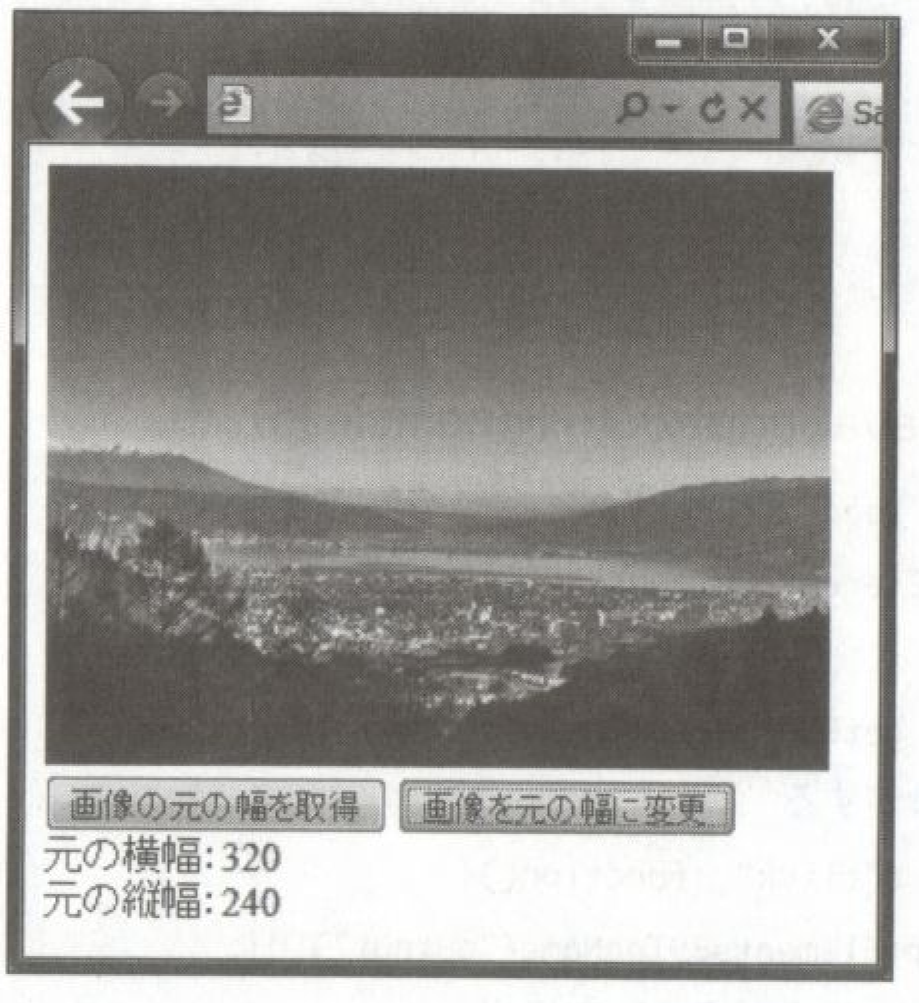
```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // img要素を読み出す
  var myImage = document.getElementsByTagName("img")[0];
  // ボタンにイベントを設定する
  btn[0].addEventListener("click", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 表示されている画像の元の横幅を読み出す
    var w = myImage.naturalWidth;
    // 表示されている画像の元の縦幅を読み出す
    var h = myImage.naturalHeight;
    // 結果を出力
    ele.innerHTML = "元の横幅：”+w+”<br>元の縦幅：”+h;
  }, false);
  // ボタンにイベントを設定する
  btn[1].addEventListener("click", function(){
    // 表示されている画像の横幅を元の幅にする
    myImage.width = myImage.naturalWidth;
```

```
    // 表示されている画像の縦幅を元の幅にする
    myImage.height = myImage.naturalHeight;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「画像の元の幅を取得」ボタンをクリックすると、画像の元の横幅と縦幅が表示されます。



「画像を元の幅に変更」ボタンをクリックすると、元の画像のサイズに変わります。



ONEPOINT 画像の元のサイズを取得するには「naturalWidth」プロパティと「naturalHeight」プロパティを使う

「Image」オブジェクトの「width」プロパティと「height」プロパティは、現在、表示されている画像のサイズを示しています。表示されている画像サイズではなく、元々の画像サイズを取得するには、「Image」オブジェクトの「naturalWidth」プロパティと「naturalHeight」プロパティを参照します。「naturalWidth」プロパティが横幅、「naturalHeight」プロパティが縦幅で、読み出された数値の単位はピクセルになります。

関連項目 ▶▶▶

● 画像の幅を設定・取得する …… p.311

SECTION-134

画像データが読み込まれた場合に処理する

ここでは、画像データが読み込まれた場合に処理する方法について解説します。

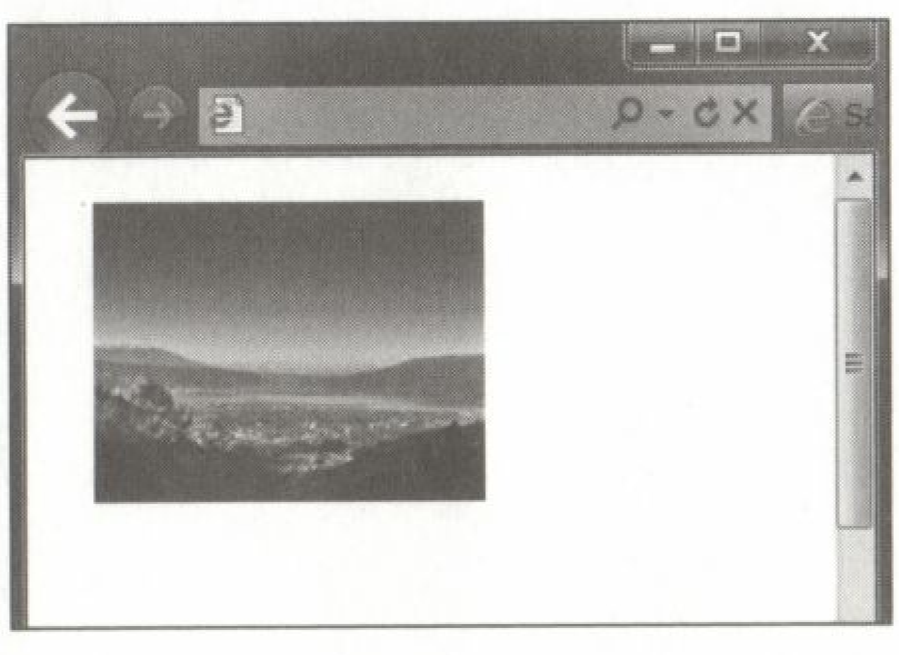
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを利用してください
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // canvas要素を読み出す
  var canvasObj = document.getElementsByTagName("canvas")[0];
  // コンテキストを読み出す
  var context = canvasObj.getContext("2d");
  // 画像を読み込む
  var myImage = new Image();
  myImage.src = "images/photo0.jpg";
  // 画像の読み込み完了時に呼び出すイベントハンドラを設定
  myImage.onload = function(){
    context.drawImage(this, 20,10);
  }
}, false);
```

ブラウザでindex.htmlを表示すると、画像データが読み込まれた後、Canvasに画像が描画されます。



ONEPOINT　画像の読み込み完了のチェックを行うには「onload」プロパティを使う

「Image」オブジェクトでは、画像が読み込まれると、「load」イベントが発生します。「Image」オブジェクトには「onload」プロパティがあり、イベントハンドラを設定しておくことで、画像読み込み完了時に処理することができます。

Canvasに画像を描画する場合は、画像データが読み込まれている必要があります。サンプルでは、画像の読み込み完了時に、Canvasに画像を指定位置に描画しています。

関連項目 ▶▶▶

- 「Image」オブジェクトを生成する …………………………………………… p.309
- 画像データを入れ替える ………………………………………………………… p.317
- 画像や映像ファイルを読み込む …………………………………………………… p.687

SECTION-135

画像データを入れ替える

ここでは、画像データを入れ替える方法について解説します。

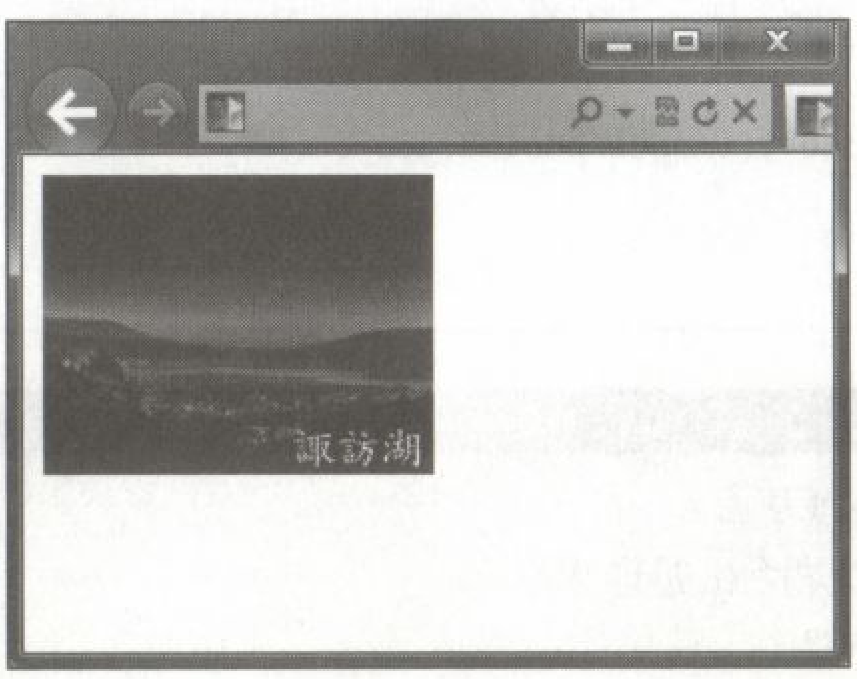
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <div>
      <img src="images/photo_gray0.jpg">
    </div>
  </body>
</html>
```

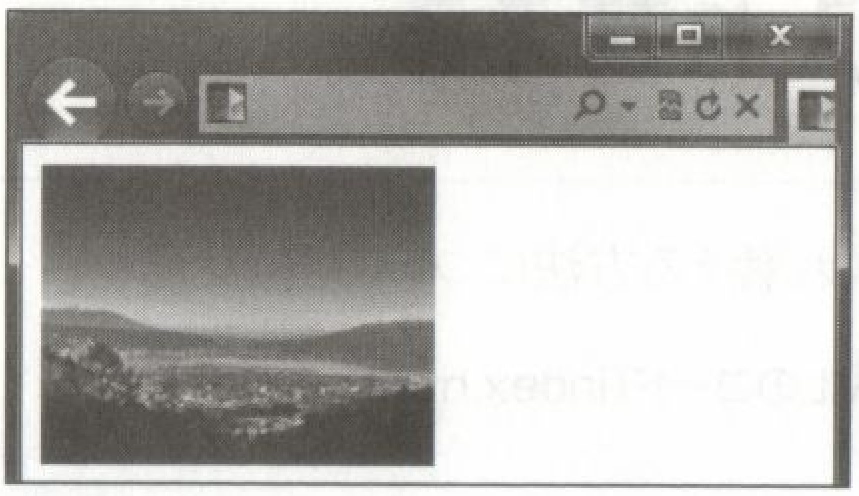
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // img要素にアクセスするために参照を変数に代入
  var myImage = document.getElementsByTagName("img")[0];
  // mouseoverイベントを割り当てる
  myImage.addEventListener("mouseover", function(){
    this.src = "images/photo0.jpg";
  }, false);
  // mouseoutイベントを割り当てる
  myImage.addEventListener("mouseout", function(){
    this.src = "images/photo_gray0.jpg";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、画像上にマウスを乗せると、画像が入れ替わります。Androidの場合は、タッチすると画像が入れ替わります。



マウスを画像から離すと、元の画像に戻ります。



ONEPOINT 画像の入れ替えを行うには「src」プロパティを使う

画像を入れ替えるには、「Image」オブジェクトの「src」プロパティに画像のURL（URI）を指定します。その際、入れ替える画像データが読み込まれていないと、瞬時に切り替わりません。低速な回線の場合は、次のようにあらかじめ、画像を読み込んでおく方法もあります。

```
(new Image()).src = "画像のURL";
```

なお、ロールオーバーによる単純な画像の入れ替えであれば、JavaScriptを利用するよりもCSSを使って入れ替える方がよい場合もあります（下記のCOLUMN参照）。

COLUMN CSSでロールオーバーを行うには

CSSでロールオーバーを行うには、「:hover」疑似クラスを利用します。次の場合、HTMLで「<div id="myImage"></div>」となっている背景画像を入れ替えます（317ページのサンプルと同様の処理）。

また、複数画像の入れ替えではなく、1枚の画像を入れ替える（CSSスプライト）場合は、「background-position」を使って表示する画像の位置を変化させることでロールオーバーを実現することができます。

```
#myImage {
  width: 160px;
  height: 120px;
  background-image: url(images/photo_gray0.jpg);
}
#myImage:hover {
  background-image: url(images/photo0.jpg);
}
```

関連項目 ▶▶▶

SECTION-136

フォーム要素にアクセスする

ここでは、フォーム要素にアクセスする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

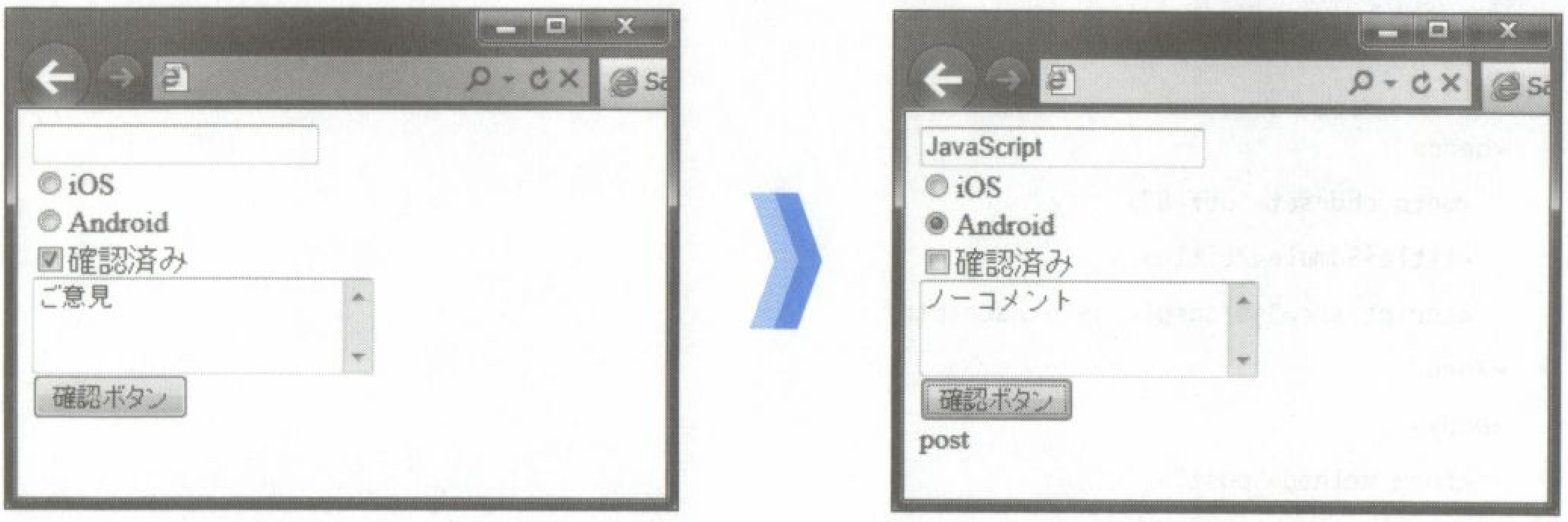
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form method="post">
      <input type="text"><br>
      <input type="radio" name="os">iOS<br>
      <input type="radio" name="os">Android<br>
      <input type="checkbox" name="myCheck" checked>確認済み<br>
      <textarea rows="3" cols="20">ご意見</textarea><br>
      <input type="button" value="確認ボタン" id="chkBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // フォームのinputボタンにgetElementById()でアクセス
  var btn = document.getElementById("chkBtn");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 最初のフォーム要素
    var f = document.forms[0];
    // フォームのmethod属性にアクセス
    ele.innerHTML = f.method +"<br>";
    // elementsで最初の要素にアクセス
    f.elements[0].value = "JavaScript";
    // 2番目のラジオボタンを選択
    f.elements["os"][1].checked = true;
    // チェックボックスの選択状態を切り替え
    f.myCheck.checked = !f.myCheck.checked;
    // テキストエリアに文字を出力
    document.querySelectorAll("textarea")[0].value = "ノーコメント";
```

```
    }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「確認ボタン」をクリックすると、フォームの「method」プロパティの値が表示され、ラジオボタンやチェックボックスの状態が変化します。また、テキストフィールドやテキストエリア内に文字が出力されます。



ONEPOINT フォームを制御するには要素にアクセスする

フォーム、および、フォーム要素にアクセスする方法は多様です。次のような方法で各要素にアクセスすることができます。

◉「form」要素

```
document.forms[番号]
document.フォーム名
```

◉form内の各要素

```
document.forms[番号].elements[番号]
document.forms[番号].elements[番号][番号]  // ラジオボタンの場合
document.forms[番号][番号]
```

◉「form」要素と各要素共通

```
// ■はHTMLの要素
■.getElementById(ID名)
■.getElementsByTagName(要素名)
■.getElementsByName(名前)
■.getElementsByClassName(要素名)
■.querySelector(セレクタ)
■.querySelectorAll(セレクタ)
```

SECTION-137

ラジオボタン/チェックボックスの状態を設定・取得する

ここでは、ラジオボタン/チェックボックスの状態を設定・取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

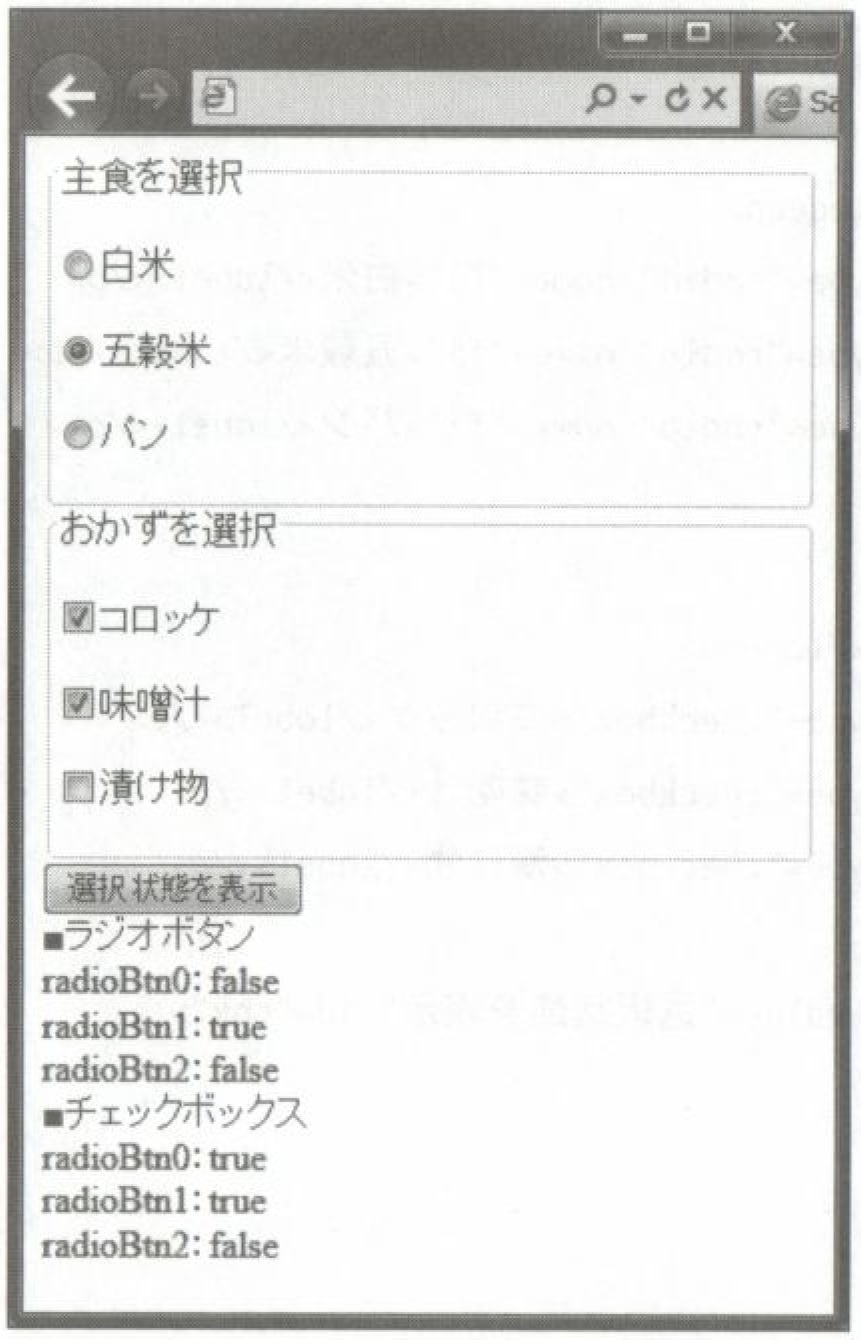
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form name="selectFoods">
      <fieldset>
      <legend>主食を選択</legend>
        <p><label><input type="radio" name="f1">白米</label></p>
        <p><label><input type="radio" name="f1">五穀米</label></p>
        <p><label><input type="radio" name="f1">パン</label></p>
      </fieldset>
      <fieldset>
      <legend>おかずを選択</legend>
        <p><label><input type="checkbox">コロッケ</label></p>
        <p><label><input type="checkbox">味噌汁</label></p>
        <p><label><input type="checkbox">漬け物</label></p>
      </fieldset>
      <input type="button" value="選択状態を表示" id="chk">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementById("chk");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    var ele = document.getElementsByTagName("output")[0];
    // ラジオボタンの選択状況を読み出し
    var radioBtn = document.forms["selectFoods"]["f1"];
    // チェックボックスの選択状態を読み出し
    var checkBtn = document.querySelectorAll("input[type=checkbox]");
    // 読み出した結果を出力
    ele.innerHTML = "■ラジオボタン<br>"
    for(var i=0; i<radioBtn.length; i++){
```

```
      ele.innerHTML += "radioBtn"+i+":"+radioBtn[i].checked+"<br>";
    }
    ele.innerHTML += "■チェックボックス<br>";
    for(var i=0; i<checkBtn.length; i++){
      ele.innerHTML += "radioBtn"+i+":"+checkBtn[i].checked+"<br>";
    }
    // デフォルトイベントを禁止
    evt.preventDefault();
    return false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「選択状態を表示」ボタンをクリックすると、ラジオボタンとチェックボックスの選択状態が表示されます。



ONEPOINT

ラジオボタンとチェックボックスの選択状態を調べるには「checked」プロパティを使う

ラジオボタンとチェックボックスの選択状態を調べるには、要素の「checked」プロパティを参照します。選択されている場合は「true」、選択されていない場合は「false」になります。「checked」プロパティは読み出しだけでなく、設定することもできます。

また、フォーム要素でチェックされているラジオボタン/チェックボックスだけを取得したい場合は、「document.querySelectorAll("input:checked")」のようにセレクタを利用する方法もあります。

関連項目 ▶▶▶

● フォーム要素にアクセスする ……………………………… p.319

SECTION-138

フォーム要素のフォーカスを設定する

ここでは、フォーム要素にフォーカスしたり、内容を選択したりする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><label>名前：
      <input type="text" id="uName" value="KF" placeholder="名前を入れてください">
      </label></p>
      <p><label>予約日：
      <input type="date">
      </label></p>
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementById("uName");
  // 最初のテキストフィールドにフォーカス
  ele.focus();
  // 入力されている文字を選択
  ele.select();
}, false);
```

ブラウザでindex.htmlを表示し、ページが読み込まれると同時に最初のテキストフィールドにフォーカスが移ります。さらに、テキストフィールド内の文字が選択状態になります。



ONEPOINT **フォーカスを設定するには「focus()」メソッドや「blur()」メソッド、「select()」メソッドを使う**

入力フォームにはさまざまなエレメントがあります。ページが読み込まれた際や入力エラーで、エラー発生場所のエレメントにフォーカスしたい場合があります。特定のエレメントにフォーカスするには、「focus()」メソッドを使います。フォーカスを外したい場合には、「blur()」メソッドを使います。なお、テキストフィールドの場合、「focus()」メソッド実行後に「select()」メソッドを実行すると、入力済みの文字を選択状態にすることができます。

関連項目 ▶▶▶

● フォーム要素にアクセスする ……………………………… p.319
● フォーカスした際に処理を行う ……………………………… p.412

SECTION-139

フォーム内容を読み出す

ここでは、フォーム内容を読み出す方法について解説します。

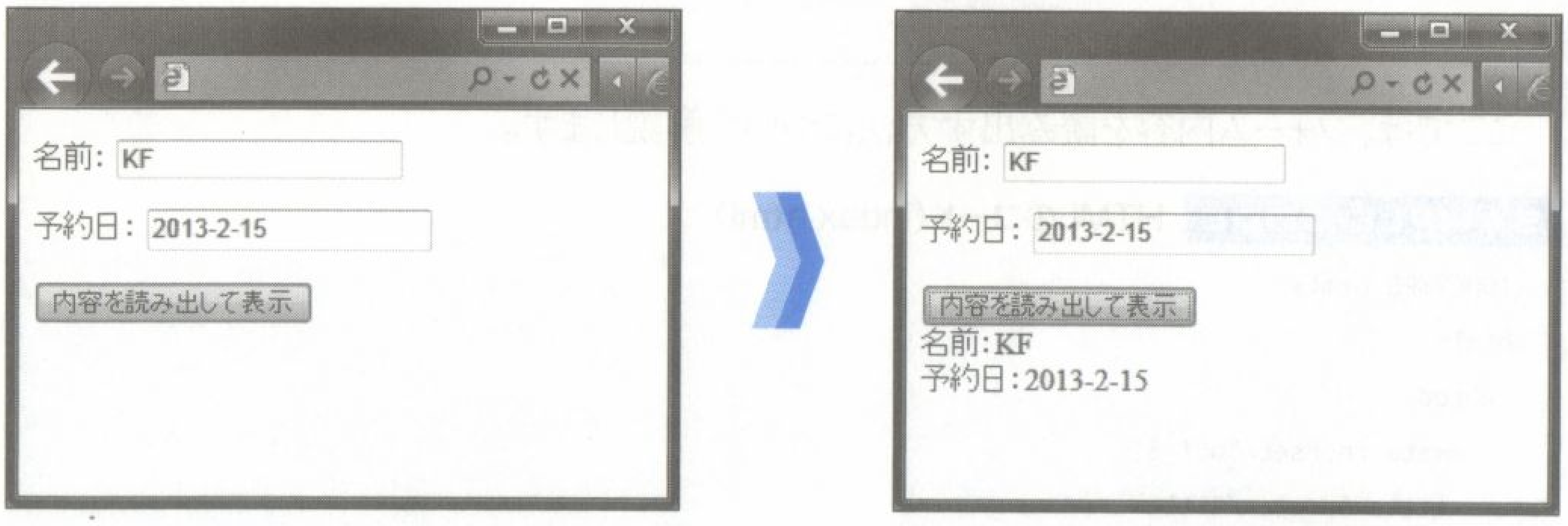
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><label>名前：
      <input type="text" id="uName" value="KF" placeholder="名前を入れてください">
      </label></p>
      <p><label>予約日：
      <input type="date" id="reservedDay">
      </label></p>
      <button>内容を読み出して表示</button>
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("button")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    var ele = document.getElementsByTagName("output")[0];
    // 名前を入力するテキストフィールドの値を読み出す
    var userName = document.getElementById("uName").value;
    // 予約日を入力するテキストフィールドの値を読み出す
    var resDay = document.getElementById("reservedDay").value;
    // 読み出した結果を出力
    ele.innerHTML = "名前 :"+userName+"<br>";
    ele.innerHTML += "予約日 :"+resDay;
    // デフォルトイベントを禁止
    evt.preventDefault();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに入力した後、「内容を読み出して表示」ボタンをクリックすると、入力した内容が画面に出力されます。



ONEPOINT　入力された内容を読み出すには「value」プロパティを使う

テキストフィールドに入力された内容を読み出すには、要素の「value」プロパティにアクセスします。テキストフィールドに数値を入れた場合でも「value」プロパティで読み出すと、数値型ではなく文字型として返されます。数値型として読み出す場合には、「valueAsNumber」プロパティを参照します。ただし、「valueAsNumber」プロパティでアクセスできるのは、「input」要素の「type」属性に「number」を指定した場合になります。他の形式を指定した場合は、「valueAsNumber」プロパティから読み出すことができません。

関連項目 ▶▶▶

- フォーム要素にアクセスする ……………… p.319
- 内容をDate型/数値型として読み出す ……………… p.327

SECTION-140

内容をDate型/数値型として読み出す

ここでは、内容をDate型/数値型として読み出す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><label>発注数：
      <input type="number" id="oNumber" value="5">
      </label></p>
      <p><label>予約日：
      <input type="date" id="reservedDay">
      </label></p>
      <button>内容を読み出して表示</button>
    </form>
    <output></output>
  </body>
</html>
```

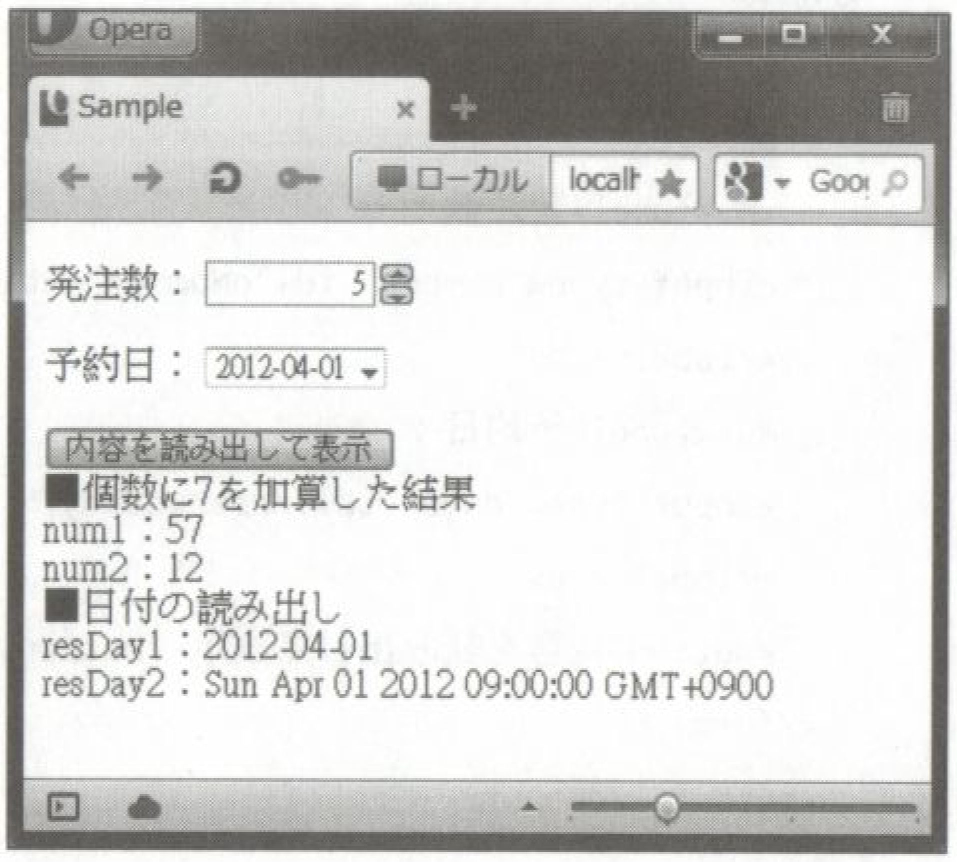
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("button")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    var ele = document.getElementsByTagName("output")[0];
    // 数値テキストフィールドをvalueで読み出し+記号で処理
    var num1 = document.getElementById("oNumber").value;
    num1 = num1 + 7;
    // 数値テキストフィールドvalueAsNumberで読み出+記号で処理
    var num2 = document.getElementById("oNumber").valueAsNumber;
    num2 = num2 + 7;
    // 日付テキストフィールドの値をvalueで読み出す
    var resDay1 = document.getElementById("reservedDay").value;
    // 日付テキストフィールドの値をvalueAsDateで読み出す
    var resDay2 = document.getElementById("reservedDay").valueAsDate;
    // 読み出した結果を出力
    ele.innerHTML = "■個数に7を加算した結果<br>"
```

```
    ele.innerHTML += "num1:"+num1+"<br>";
    ele.innerHTML += "num2:"+num2+"<br>";
    ele.innerHTML += "■日付の読み出し<br>";
    ele.innerHTML += "resDay1:"+resDay1+"<br>";
    ele.innerHTML += "resDay2:"+resDay2;
    // デフォルトイベントを禁止
    evt.preventDefault();
    return false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィールドに入力した後、「内容を読み出して表示」ボタンをクリックすると、入力した数値に「7」を加算した結果と、設定した日付が画面に出力されます。





ONEPOINT 数値型として読み出すには「valueAsNumber」プロパティ、日付型として読み出すには「valueAsDate」プロパティを使う

HTML5の入力フォームには、数値専用入力と日付専用入力を行うフィールドがあります。これらのフィールドの内容は、従来通り、「value」プロパティを使って読み出すことができます。ただし、「value」プロパティを使って読み出した場合、内容が数値であっても文字型として読み出されます。数値型として読み出す場合には、「valueAsNumber」プロパティを使います。ただし、「type="number"」以外では、「valueAsNumber」プロパティは正しく機能しません。

「type="date"」「type="datetime"」「type="datetime-local"」が指定されている場合、「valueAsDate」プロパティを使うと、日付を示す文字列として読み出すことができます。

関連項目 ▶▶▶

● フォーム要素にアクセスする ………… p.319

SECTION-141

フォームへの入力を禁止・許可する

ここでは、フォームへの入力を禁止・許可する方法について解説します。

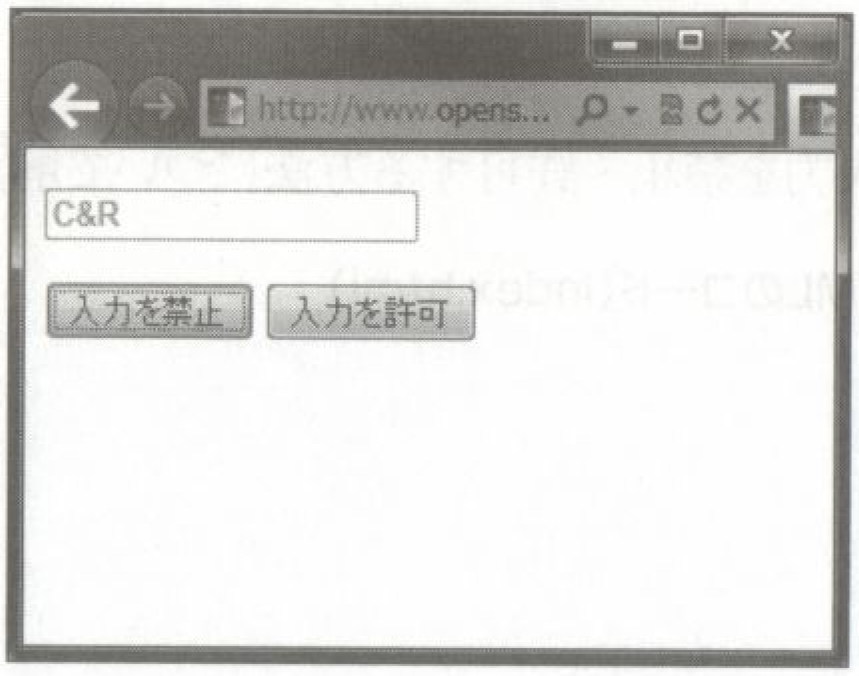
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="text" id="uName" placeholder="名前を入れてください"></p>
      <input type="button" value="入力を禁止" id="disBtn">
      <input type="button" value="入力を許可" id="enaBtn">
    </form>
  </body>
</html>
```

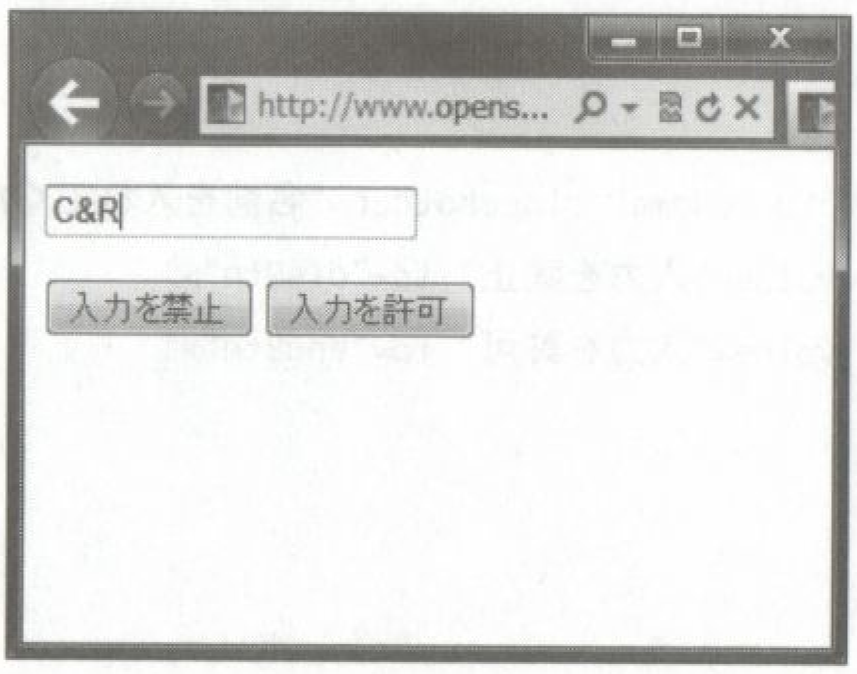
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var myField = document.getElementById("uName");
  var dBtn = document.getElementById("disBtn");
  var eBtn = document.getElementById("enaBtn");
  // 入力禁止ボタンにイベントを割り当て
  dBtn.addEventListener("click", function(evt){
    // フィールドへの入力を禁止
    myField.disabled = true;
  }, false);
  // 入力許可ボタンにイベントを割り当て
  eBtn.addEventListener("click", function(evt){
    // フィールドへの入力を許可
    myField.disabled = false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「入力を禁止」ボタンをクリックすると、テキストフィールドへの入力ができなくなります。



「入力を許可」ボタンをクリックすると、テキストフィールドへの入力が可能になります。



ONEPOINT テキストフィールドへの入力を禁止するには「disabled」プロパティを使う

テキストフィールドへの入力を禁止するには、「disabled」プロパティに「true」を設定します。「false」を設定すると、入力が可能な状態になります。「disabled」プロパティはテキストフィールドだけでなく、ラジオボタンやチェックボックス、セレクトメニューに対しても有効です。

なお、「form」要素には「disabled」プロパティはないため、一括してすべての要素の入力を禁止することはできません。

関連項目 ▶▶▶

● フォーム要素にアクセスする ……… p.319

SECTION-142

セレクトメニューで選択された項目を取得する

ここでは、セレクトメニューで選択された項目を取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

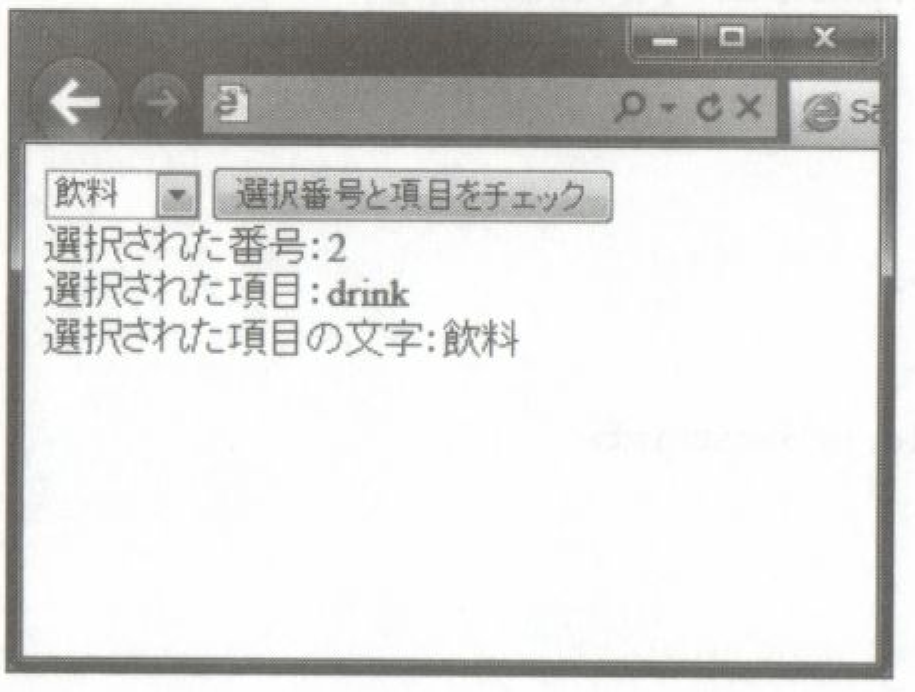
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <select id="food">
        <option value="rice">白米</option>
        <option value="pasta">パスタ</option>
        <option value="drink">飲料</option>
        <option value="beer">ビール</option>
      </select>
      <input type="button" value="選択番号と項目をチェック" id="chk">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // セレクトメニューの要素を読み出し
  var mySelMenu = document.getElementById("food");
  // ボタン要素の読み出し
  var btn = document.getElementById("chk");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // セレクトメニューの選択番号を取得
    var no = mySelMenu.selectedIndex;
    // セレクトメニューの選択項目を取得
    var item = mySelMenu.value;
    // セレクトメニューの選択項目の文字を取得
    var text = mySelMenu.options[no].text;
    // 結果を出力
    ele.innerHTML = "選択された番号 :"+no+"<br>";
    ele.innerHTML += "選択された項目 :"+item+"<br>";
```

```
    ele.innerHTML += "選択された項目の文字 :"+text+"<br>";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「選択番号と項目をチェック」をクリックすると、セレクトメニューで選択されている番号や項目などが表示されます。



ONEPOINT

選択された項目の情報を取得するには「selectedIndex」プロパティや「value」プロパティ、「text」プロパティを使う

セレクトメニューで選択された項目に関しては、いくつかの情報を取得することができます。まず、選択された番号に関しては、セレクトメニューオブジェクトの「selectedIndex」プロパティに入っています。選択された項目の値(「option」要素の「value」の内容)は、セレクトメニューオブジェクトの「value」プロパティに入っています。

選択された値ではなく、「options」要素で指定されている文字(実際にメニューに表示されている文字)を取得するには、選択された「options」配列の「text」プロパティを使います。

関連項目 ▶▶▶

● フォーム要素にアクセスする ……………………………… p.319

SECTION-143

フォーム内容を初期化する

ここでは、フォーム内容を初期化(リセット)する方法について解説します。

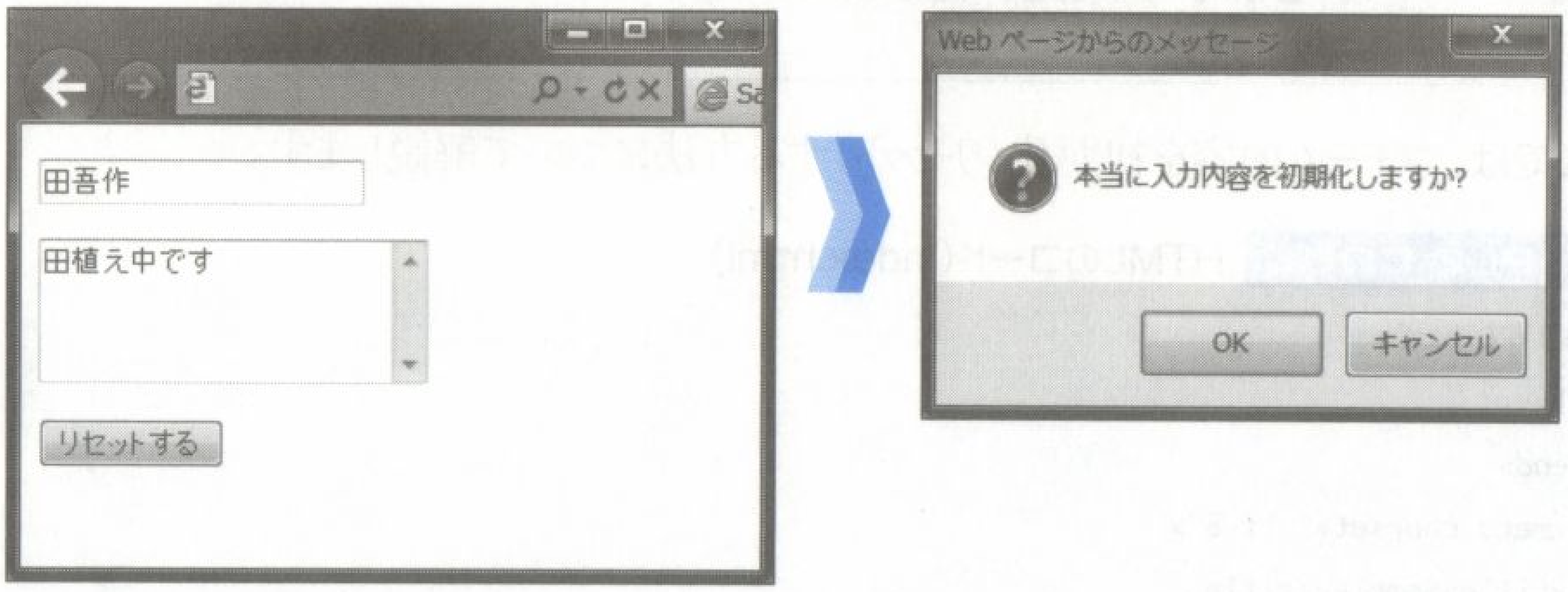
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form name="enq">
      <p><input type="text" value="与作"></p>
      <p><textarea rows="4" cols="20">伐採中です</textarea></p>
      <input type="button" value="リセットする" id="resetBtn">
    </form>
  </body>
</html>
```

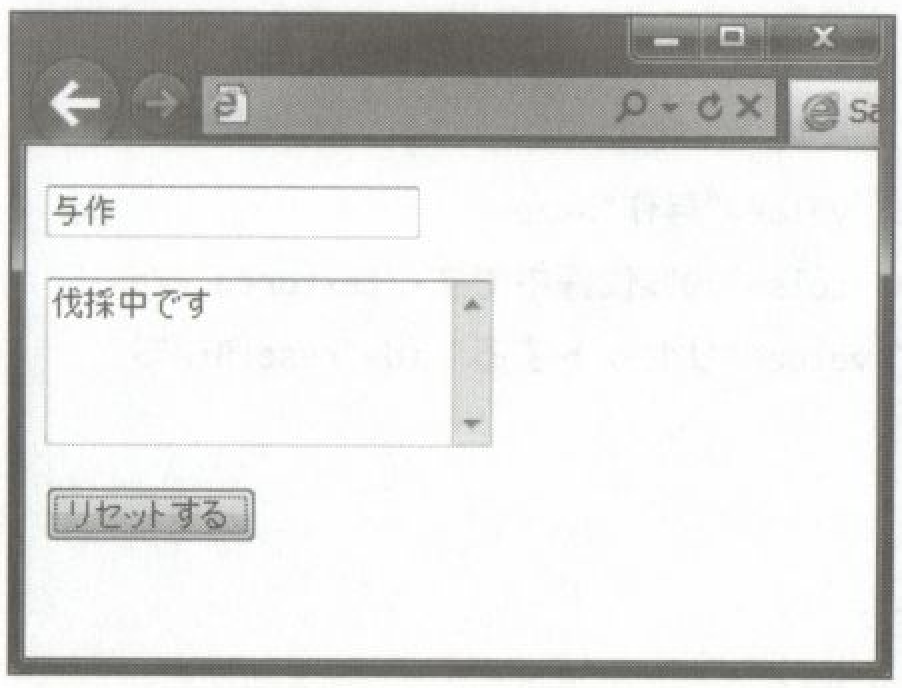
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementById("resetBtn");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 削除するか確認する
    var flag = confirm("本当に入力内容を初期化しますか?");
    if (flag){
      // ダイアログで「OK」がクリックされたのでリセットを実行
      this.form.reset();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドとテキストエリアに文字を入力して、「リセットする」ボタンをクリックすると、確認のダイアログが表示されるます。



「OK」ボタンをクリックすると、フォームの内容が初期化(リセット)されます。



ONEPOINT　フォーム内容を初初期化するには「reset()」メソッドを使う

フォーム内容を一括して初期化(リセット)するには、「reset()」メソッドを使います。「reset()」メソッドは、「form」要素で「<input type="reset">」とした場合と同じ役割になります。初期化されるのは、フォーム要素内のすべてのエレメントになります。

関連項目 ▶▶▶

●リセット時・送信時に処理する……………………………………………………p.429

SECTION-144

数値専用のフィールドで値を増減する

ここでは、数値専用のフィールドで値を増減する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

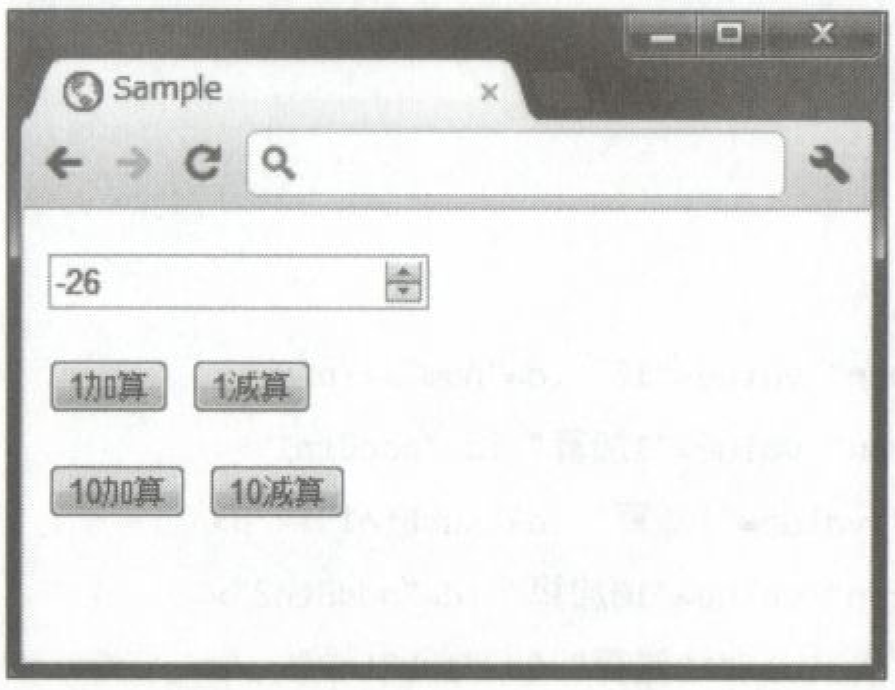
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form name="enq">
      <p><input type="number" value="10" id="num"></p>
      <p><input type="button" value="1加算" id="addBtn1">
      <input type="button" value="1減算" id="subBtn1"></p>
      <p><input type="button" value="10加算" id="addBtn2">
      <input type="button" value="10減算" id="subBtn2"></p>
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // type="number"の要素を読み出し
  var orderNumber = document.getElementById("num");
  // ボタン要素を読み出し
  var addbtn1 = document.getElementById("addBtn1");
  var addbtn2 = document.getElementById("addBtn2");
  var subbtn1 = document.getElementById("subBtn1");
  var subbtn2 = document.getElementById("subBtn2");
  // 1加算ボタンにイベントを割り当て
  addbtn1.addEventListener("click", function(evt){
    // 1を加算する
    orderNumber.stepUp();
  }, false);
  // 10加算ボタンにイベントを割り当て
  addbtn2.addEventListener("click", function(evt){
    // 10を加算する
    orderNumber.stepUp(10);
  }, false);
  // 1減算ボタンにイベントを割り当て
  subbtn1.addEventListener("click", function(evt){
```

```
      // 1を減算する
      orderNumber.stepDown();
    }, false);
    // 10減算ボタンにイベントを割り当て
    subbtn2.addEventListener("click", function(evt){
      // 10を減算する
      orderNumber.stepDown(10);
    }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、フィールドの値が加算・減算されます。



ONEPOINT ボタンで値を増減させるには「stepUp()」メソッドと「stepDown()」メソッドを使う

数値だけを扱う場合には、「input」要素に「type="number"」を指定します。このように指定した場合、ボタンで値を増減させるには「stepUp()」メソッドと「stepDown()」メソッドを使います。「stepUp()」メソッドは値を増やし、「stepDown()」メソッドは値を減らします。パラメータを何も指定しない場合は、1ずつ増減させることができます。パラメータに値を指定した場合は、その値の分だけ、増減します。

関連項目 ▶▶▶

● フォーム要素にアクセスする …… p.319

SECTION-145

非同期通信オブジェクトを生成する

ここでは、非同期通信オブジェクトを生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>非同期通信処理</h1>
    <form method="get">
      <input type="button" value="読み込み" id="loadBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

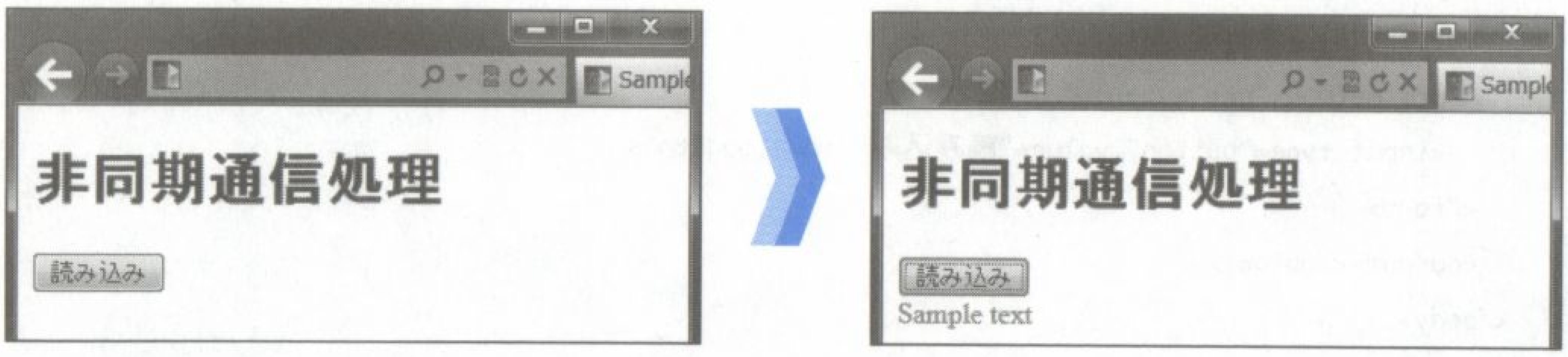
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 読み込みボタンにイベントを割り当て
  document.getElementById("loadBtn").addEventListener("click", function(evt){
    // 非同期通信オブジェクト作成
    var xhr = createXMLHttpRequest();
    // 読み込み完了時のイベントハンドラを設定
    xhr.onload = function(){
      // 読み込んだテキストを出力
      ele.innerHTML = xhr.responseText;
    }
    // 同一ドメインのURL上のファイルを指定
    xhr.open("get", "./data.html", true);
    xhr.send(null);
  }, false);
}, false);
// http通信用、IE共通関数
function createXMLHttpRequest(){
  var XMLhttpObject = null;
  try{ XMLhttpObject = new XMLHttpRequest();
  }catch(e){
    try{ XMLhttpObject = new ActiveXObject("Msxml2.XMLHTTP");
```

```
    }catch(e){
      try{ XMLhttpObject = new ActiveXObject("Microsoft.XMLHTTP");
      }catch(e){ return null; }
    }
  }
  return XMLhttpObject;
}
```

HINT

非同期通信は、ローカルディスク上では動作しません。ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、非同期通信用オブジェクトが生成され、サーバーからHTMLファイルが読み込まれ表示されます。



ONEPOINT　非同期通信を行うには「XMLHttpRequest」オブジェクトを使う

IE9以降、および、Firefox、Safari、Google Chrome、Operaなどでは、「XMLHttpRequest」オブジェクトが非同期通信を行うオブジェクトになっています。IE9よりも前のIEではActiveXを利用して非同期通信用オブジェクトを生成する必要があります。

また、「readyState」プロパティと「status」プロパティを利用して通信状態に応じて処理を行うこともできます。「readyState」プロパティには、次の0～4までの数値が入ります。

値	説明
0	未初期化
1	接続開始
2	リクエスト受信
3	処理中
4	完了

「status」プロパティには、サーバーからのステータス番号が入ります。「200」なら正常、「404」ならファイルが見つからないことになります。ステータスコードについては、Wikipedia(次のURL)などを参照してください。

- HTTPステータスコード - Wikipedia

URL http://ja.wikipedia.org/wiki/HTTPステータスコード

関連項目 ▶▶▶

● サーバーからテキストデータやXMLデータを受信する …………………………… p.339

SECTION-146

サーバーからテキストデータやXMLデータを受信する

ここでは、サーバーからテキストデータやXMLデータを受信する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>非同期通信で読み込む</h1>
    <form method="get">
      <input type="button" value="テキストファイルを読み込み" id="loadTextBtn">
      <input type="button" value="XMLファイルを読み込み" id="loadXMLBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // テキストファイルを読み込みボタンにイベントを割り当て
  document.getElementById("loadTextBtn").addEventListener("click", function(evt){
    // 非同期通信オブジェクト作成
    var xhr = new XMLHttpRequest();
    // 読み込み完了時のイベントハンドラを設定
    xhr.onload = function(){
      // 読み込んだテキストを出力
      ele.innerHTML = xhr.responseText;
    }
    // 同一ドメインのURL上のファイルを指定
    xhr.open("get", "./data.txt", true);
    xhr.send(null);
  }, false);
  // XMLファイルを読み込みボタンにイベントを割り当て
  document.getElementById("loadXMLBtn").addEventListener("click", function(evt){
    // 非同期通信オブジェクト作成
    var xhr = new XMLHttpRequest();
    // 読み込み完了時のイベントハンドラを設定
    xhr.onload = function(){
```

```
        // 読み込んだXMLを解析して出力
        var xml = xhr.responseXML;
        // <name>タグをピックアップ
        var nameList = xml.getElementsByTagName("name");
        // <name>タグの数だけ繰り返して表示
        for(var i=0; i<nameList.length; i++){
          // 名前を読み出し
          var uname = nameList[i].firstChild.nodeValue;
          ele.innerHTML += uname + "<br>";
        }
      }
      // 同一ドメインのURL上のファイルを指定
      xhr.open("get", "./data.xml", true);
      xhr.send(null);
    }, false);
}, false);
```

SAMPLE CODE XML(data.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <user><name>Furuhata</name><power>2800</power></user>
  <user><name>Yoshinari</name><power>1780</power></user>
  <user><name>Yamada</name><power>1580</power></user>
  <user><name>Takahashi</name><power>1580</power></user>
</list>
```

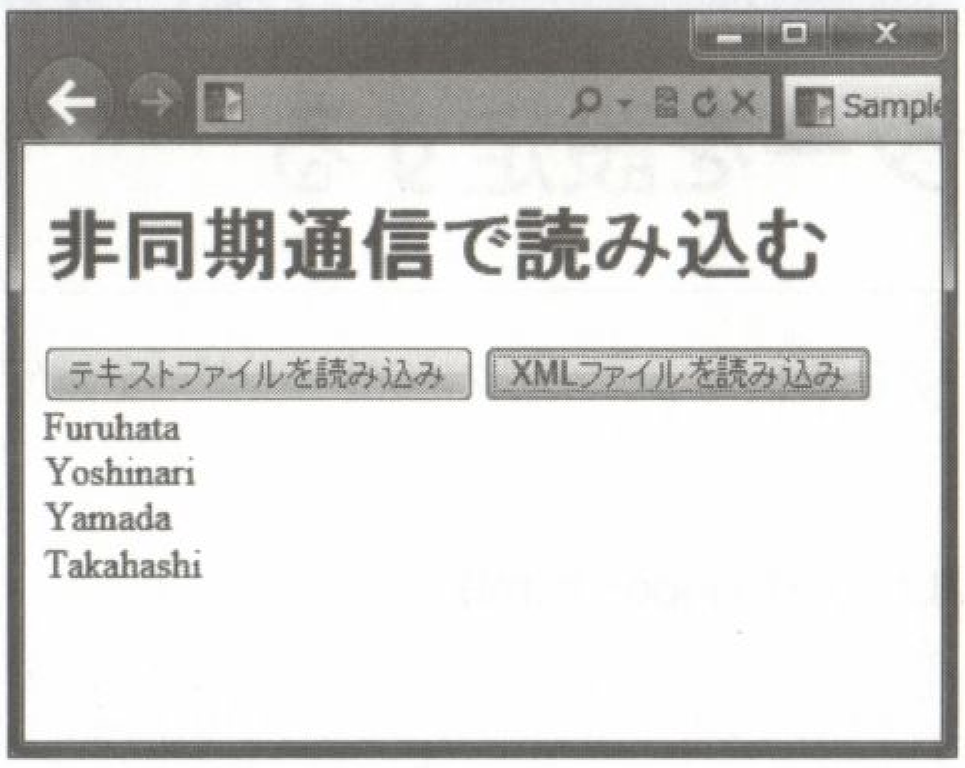
HINT

非同期通信は、ローカルディスク上では動作しません。ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、「テキストファイルを読み込み」ボタンをクリックすると、テキストファイルが読み込まれて内容が表示されます。



「XMLファイルを読み込み」ボタンをクリックすると、XMLファイルが読み込まれて<name>タグの最初のノードが表示されます。



ONEPOINT **サーバーからテキストデータを受信するには「responseText」プロパティ、XMLデータを受信するには「responseXML」プロパティを使う**

テキストデータは、データ受信後に「responseText」プロパティを参照します。テキストデータに限らず、HTMLファイルでもJSON形式のファイルでも読み込むことができます。

XMLファイルの場合、「responseXML」プロパティにXMLデータが入ります。XML DOMになるため、「getElementsByTagName()」メソッドや「nodeValue」プロパティを使ってノードにアクセスすることができます。

関連項目 ▶▶▶

● 非同期通信オブジェクトを生成する p.337

リクエストヘッダーを設定する

ここでは、サーバーにデータを送信する際のリクエストヘッダーを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>リクエストヘッダー書き換え</h1>
    <form method="get">
      <input type="button" value="テキストファイルを読み込み" id="loadBtn">
    </form>
    <output></output>
  </body>
</html>
```

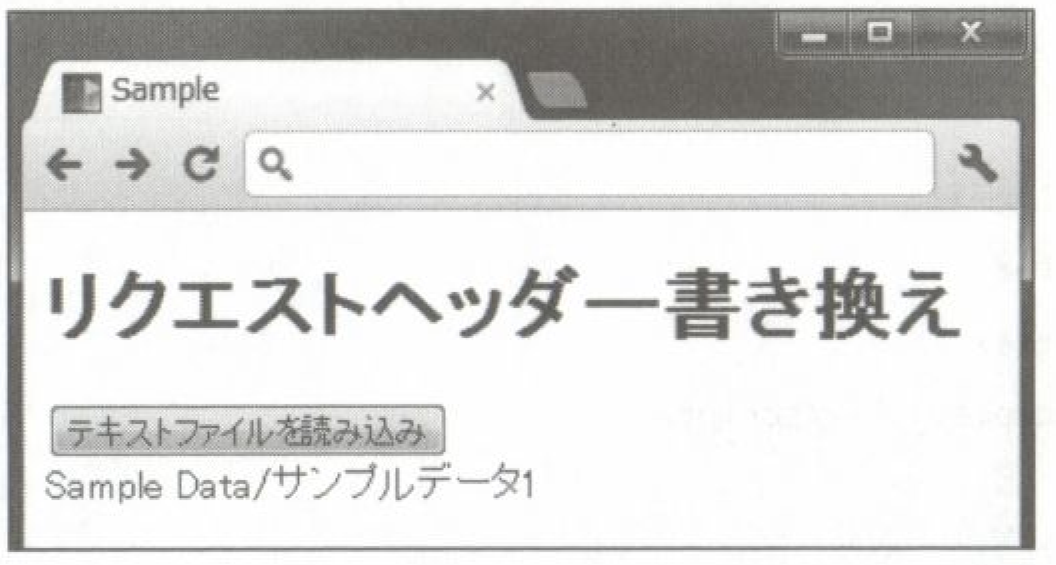
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // XMLファイルを読み込みボタンにイベントを割り当て
  document.getElementById("loadBtn").addEventListener("click", function(evt){
    // 非同期通信オブジェクト作成
    var xhr = new XMLHttpRequest();
    // 読み込み完了時のイベントハンドラを設定
    xhr.onload = function(){
      // 読み込んだ内容を出力
      ele.innerHTML = xhr.responseText;
    }
    // 同一ドメインのURL上のファイルを指定
    xhr.open("get", "./data.txt", true);
    // キャッシュしないようにする(HTTP1.1)
    xhr.setRequestHeader("Cache-Control", "no-cache");
    // リクエスト送信
    xhr.send(null);
  }, false);
}, false);
```

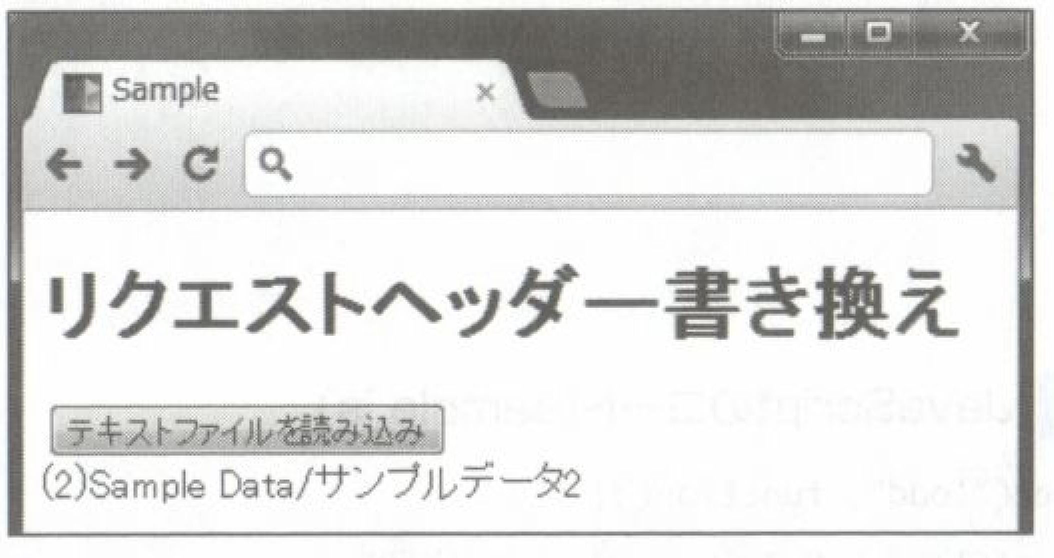
HINT

非同期通信は、ローカルディスク上では動作しません。ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、「テキストファイルを読み込み」ボタンをクリックすると、テキストファイルが読み込まれて内容が表示されます。



テキストファイルの内容を更新し、再度、ボタンをクリックすると、「setRequestHeader()」メソッドでキャッシュしないようにしているため、最新内容が読み込まれます。



ONEPOINT リクエストヘッダーを設定するには「setRequestHeader()」メソッドを使う

リクエストヘッダーを設定するには、「setRequestHeader()」メソッドを使います。最初のパラメータにヘッダー名、2番目のパラメータに値を設定します。リクエストヘッダーで指定できる項目と内容は、次のWebページを参照してください。

- RFC2616

URL http://www.ietf.org/rfc/rfc2616.txt

COLUMN IE6～9でキャッシュデータが利用されるのを回避する方法

IE6～9の場合は、次のようにすることで、キャッシュデータが利用されるのを回避できます。なお、日付は扱うファイルの更新日時に合わせて変更してください。

```
var specDay = (new Date("2012/1/1")).toGMTString();
xhr.setRequestHeader("If-Modified-Since", specDay);
```

また、HTTP1.0にも対応する必要がある場合は、「setRequestHeader("Pragma", "no-cache")」としてください。

SECTION-148

MIMEタイプを設定する

ここでは、MIMEタイプを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>MIMEタイプ書き換え</h1>
    <form method="get">
      <input type="button" value="テキストファイルを読み込み" id="loadBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // XMLファイルを読み込みボタンにイベントを割り当て
  document.getElementById("loadBtn").addEventListener("click", function(evt){
    // 非同期通信オブジェクト作成
    var xhr = new XMLHttpRequest();
    // 読み込み完了時のイベントハンドラを設定
    xhr.onload = function(){
      // 読み込んだ内容を出力
      ele.innerHTML = xhr.responseText;
    }
    // 同一ドメインのURL上のファイルを指定
    xhr.open("get", "./data.txt", true);
    if (xhr.overrideMimeType){
      xhr.overrideMimeType("text/plain;charset=shift_jis");
    }
    xhr.send(null);
  }, false);
}, false);
```

HINT

非同期通信は、ローカルディスク上では動作しません。ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、「テキストファイルを読み込み」ボタンをクリックすると、Shift JISコードのテキストファイルが読み込まれて内容が表示されます。



MIMEタイプを上書きしない場合は、文字化けして表示されます。



ONEPOINT **MIMEタイプを設定するには「overrideMimeType()」メソッドを使う**

非同期通信でサーバーから受信したデータのMIMEタイプを上書きするには、「overrideMimeType()」メソッドを使います。これによって、XMLデータや文字コードによって文字化けしてしまう現象を解消することができます。

なお、「overrideMimeType()」メソッドは、IE6～9には実装されていないため、そのまま呼び出すとエラーになります。

SECTION-149

デバッグコンソールに文字を出力する

ここでは、デバッグコンソールに文字を出力する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

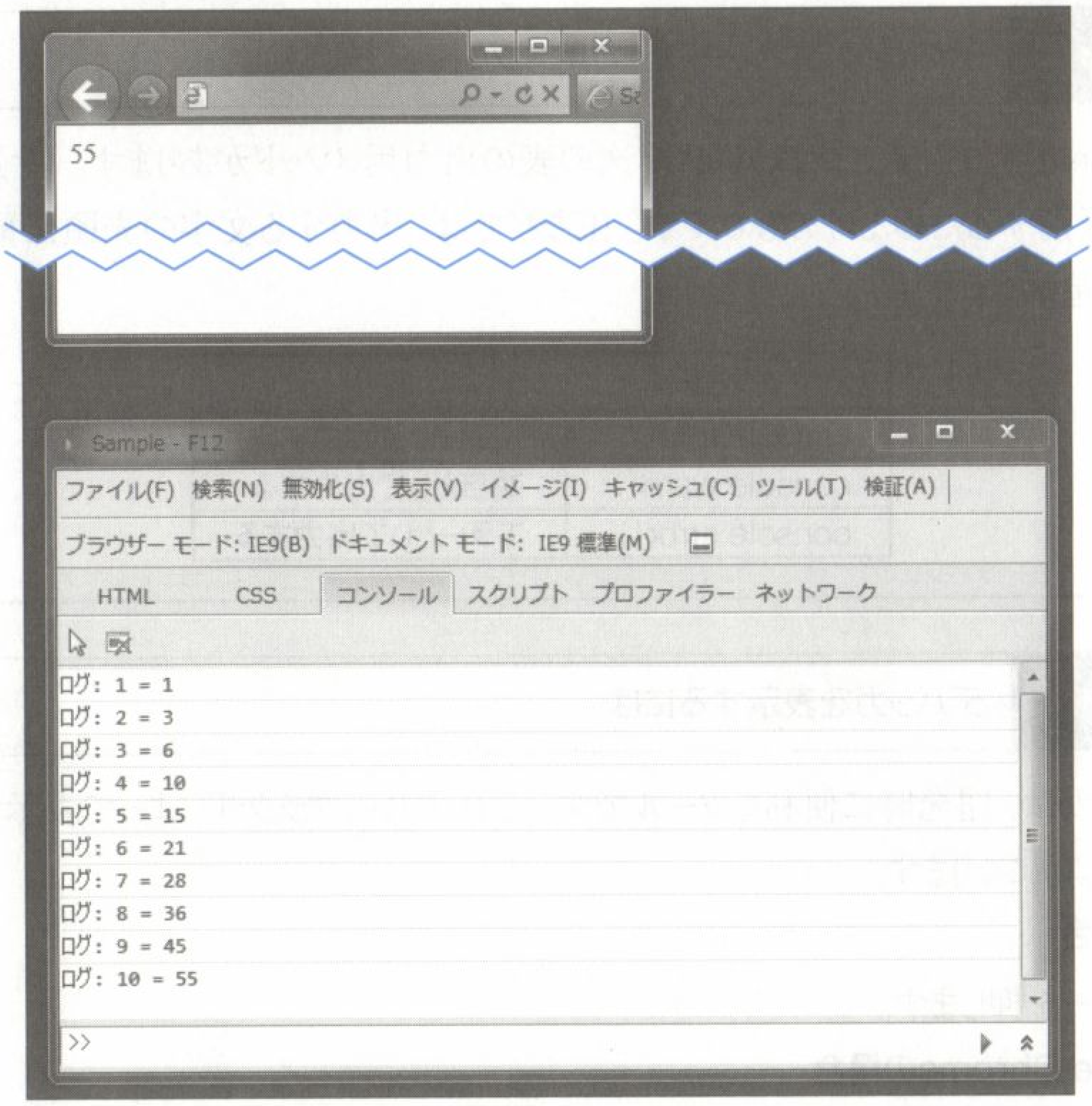
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 1から10までを加算
  var total = 0;
  for(var i=1; i<=10; i++){
    total = total + i;
    // デバッグコンソールに出力
    console.log(i+" = "+total);
  }
  ele.innerHTML = total;
}, false);
```

HINT

IEの場合は、開発者ツールを表示してから、index.htmlを表示してください。

ブラウザでindex.htmlを表示すると、デバッグコンソールに文字が表示されます。



ONEPOINT　デバッガのコンソールに出力するには「console.log()」メソッドを使う

現在のブラウザには、標準でデバッガが入っています。デバッガにはデバッグ用のコンソールがあり、JavaScriptでは「console.log()」メソッドを使って出力することができます。パラメータは「,」(カンマ)で区切って列記することができますが、出力結果はブラウザによって若干、異なる場合があります(IEでは複数のパラメータを指定した場合、出力結果が半角空白で区切られない)。

COLUMN　「console.log()」メソッドの出力結果について

デバッグコンソールには、「log()」メソッドを使って出力した文字が表示されます。このとき、変数の内容を出力することで、プログラムのミスを発見しやすくなります。Firefoxの標準のデバッガ以外では、「コンソール」タブを選択すると「log()」メソッドで出力された結果が表示されます。Firefoxの標準のデバッガでは、メニューからWebコンソールを選択すると表示されます。

COLUMN　IEの場合

IEの場合、開発者ツール(デバッグツール)を表示していないと「console」オブジェクトが存在しないため、そのまま、「console.log()」とすると、エラーになります。

COLUMN **その他のコンソール出力用メソッド**

「console.log()」メソッド以外にも、次の表の出力用メソッドがあります。いずれも、出力時に区別するためだけのものです。IEなどでは、出力する文字の先頭に警告アイコンなどが追加されます。

メソッド	説明
console.info()	情報として出力する
console.warn()	警告として出力する
console.error()	エラーとして出力する

COLUMN **デバッガを表示するには**

デバッガは、開発時に便利なツールです。それぞれのブラウザによって表示する方法は、次のようになります。

▶IEの場合

F12キーを押します。

▶Google Chromeの場合

「設定」ボタンから「ツール」→「JavaScriptコンソール」を選択します。

▶Firefoxの場合

Windowsの場合は「Firefox」→「Web開発」→「Webコンソール」を選択します。Macの場合は「ツール」メニューから「Web開発」→「Webコンソール」を選択します。

▶Safariの場合

環境設定の詳細で「メニューバーに"開発"メニューを表示」のチェックをONにした後、「開発」メニューから「エラーコンソールを表示」を選択します。

▶iPhone/iPadのSafariの場合

ホーム画面で「設定」をタップし、Safariの項目をタップします。その後、一番下の詳細をタップし、「デバッグコンソール」を「オン」にします。

▶Operaの場合

「ツール」メニューから「詳細ツール」→「Opera Dragonfly」を選択します。

関連項目 ▶▶▶

- デバッグコンソールにオブジェクトの内容を出力する ………… p.349
- 処理にかかった時間を詳細に把握する ………… p.351
- 呼び出し元の関数を把握する ………… p.353

SECTION-150

デバッグコンソールにオブジェクトの内容を出力する

ここでは、デバッグコンソールにオブジェクトの内容を出力する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p>デバッグコンソールを参照してください</p>
  </body>
</html>
```

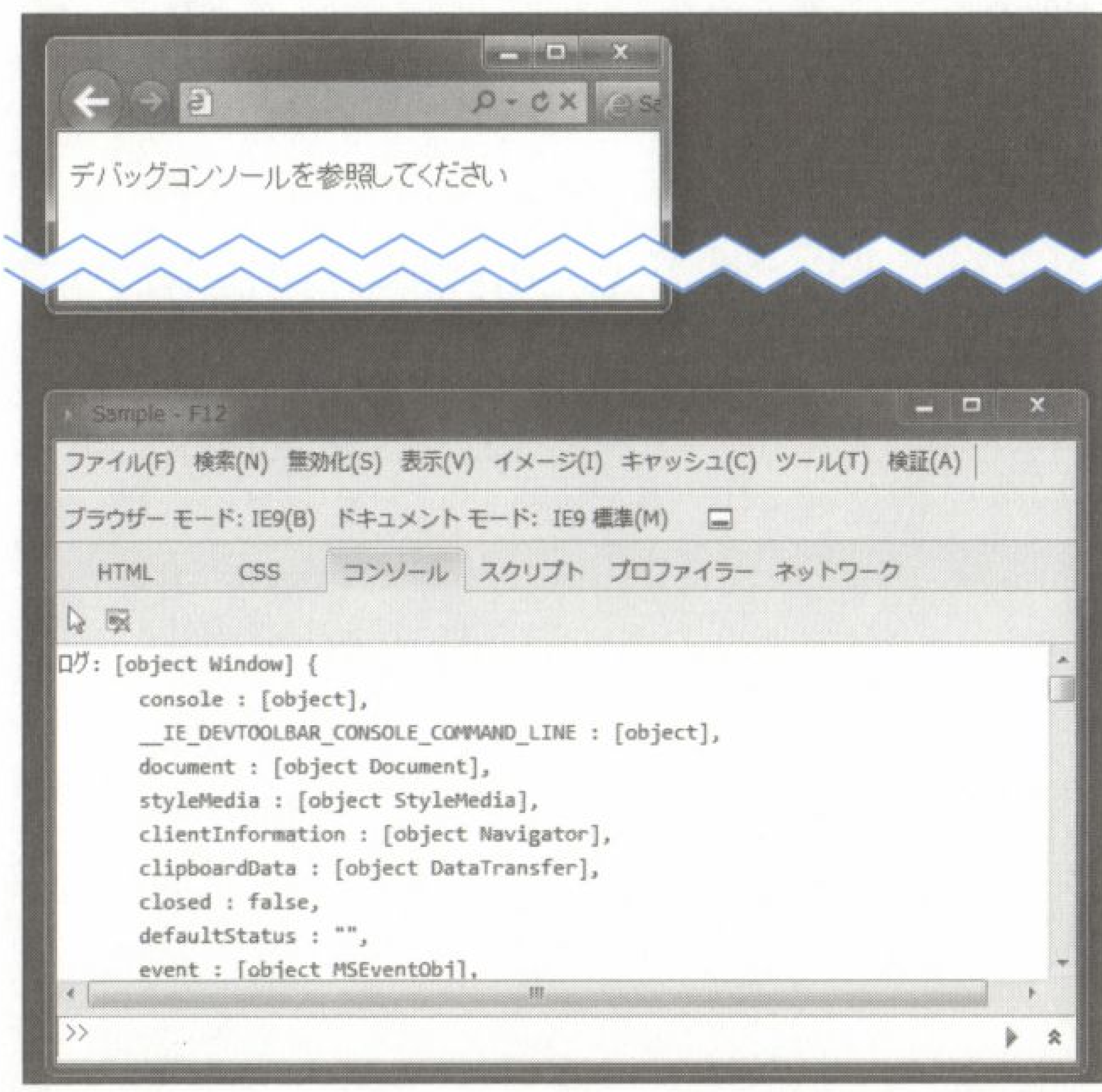
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // デバッグコンソールにwindowオブジェクトを出力
  console.dir(window);
}, false);
```

HINT

IEの場合は、開発者ツールを表示してから、index.htmlを表示してください。

ブラウザでindex.htmlを表示すると、デバッグコンソールにオブジェクトの内容が表示されます。



ONEPOINT オブジェクトの中身を確認するには「console.dir()」メソッドを使う

JavaScriptはオブジェクト指向言語であり、非常に多くのオブジェクトを扱います。標準で用意されているものだけでなく、プログラマがオブジェクトを作成することもできます。

その際、オブジェクトのプロパティなどがどのような状態になっているかを確認したい場合があります。確認するときに便利なのが「console.dir()」メソッドです。「console.dir()」メソッドのパラメータにオブジェクトを指定すると、デバッグコンソールにオブジェクトの内容が展開できる状態で出力されます。これによってオブジェクトの内容がどのようになっているか把握することができ、デバッグの手助けになります。

なお、オブジェクトにはプロパティとメソッドがあり、デバッグコンソールには「プロパティ名:内容」、または、「メソッド名:内容」の形式で出力されます。

COLUMN 「console.dir()」メソッドの出力結果について

「console.dir()」メソッドの出力結果は、Firefoxの標準のデバッガ以外では、「コンソール」タブを選択すると表示されます。Firefoxの標準のデバッガでは、メニューからWebコンソールを選択すると表示されます。

関連項目 ▶▶▶

- デバッグコンソールに文字を出力する ……… p.346
- 処理にかかった時間を詳細に把握する ……… p.351
- 呼び出し元の関数を把握する ……… p.353

SECTION-151

処理にかかった時間を詳細に把握する

ここでは、デバッガを利用して処理にかかった時間を詳細に把握する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p>デバッグコンソールを参照してください</p>
  </body>
</html>
```

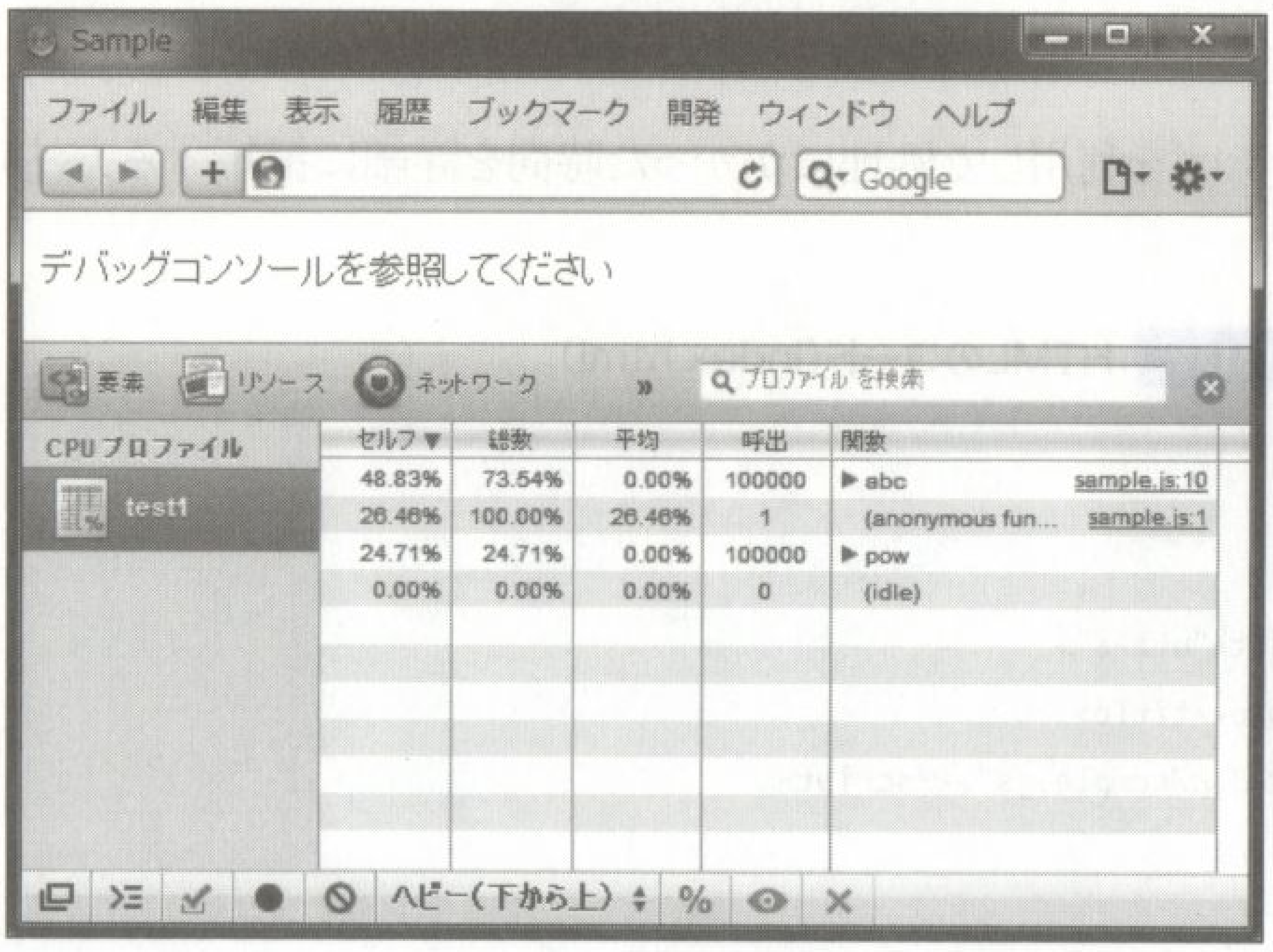
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 処理時間の計測を開始
  console.profile("test1");
  // 計測する処理を記述
  for(var a=[], i=0; i<100000; i++){
    a[i] = i * i + i / 0.1;
    var n = abc(i);
  }
  // 関数
  function abc(n){ return Math.pow(2, n); }
  // 処理時間の計測を終了
  console.profileEnd("test1");
}, false);
```

HINT

Firefoxでは、Firebugを使うことで処理にかかった時間を調べることができます。

ブラウザでindex.htmlを表示すると、デバッグコンソールに関数の処理にかかった時間などが詳細に表示されます。



ONEPOINT 処理時間を詳しく計測するにはプロファイルを利用する

特定の関数や処理にどのくらい時間がかかっているか計測するには、プロファイルを利用します。計測したい範囲の最初に「console.profile()」メソッドを指定し、範囲の最後に「console.profileEnd()」メソッドを指定します。複数箇所で計測を行う場合には、「console.profile()」メソッドのパラメータに名前を指定しておくことができます。なお、1000ミリ秒より小さい時間は計測できません。

COLUMN 「console.profile()」メソッドの出力結果について

Google ChromeとSafariの場合、「console.profile()」メソッドの出力結果については、デバッガの「Profiles」タブの「test1」をクリックすると表示されます。Firefoxでは Firebugを使い、「プロファイル」タブをクリックすると結果が表示されます。なお、「test1」が表示されていない場合は、ブラウザの再読み込みボタンをクリックしてください。

関連項目 ▶▶▶

- デバッグコンソールに文字を出力する ……… p.346
- デバッグコンソールにオブジェクトの内容を出力する ……… p.349
- 呼び出し元の関数を把握する ……… p.353

SECTION-152

呼び出し元の関数を把握する

ここでは、デバッガを利用して実行している関数が、どこから呼び出されたのかを把握する方法について解説します。

SAMPLE CODE HTML(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <p>デバッグコンソールを参照してください</p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 計測する処理を記述
  var n = abc(2);
  // 関数
  function abc(n){
    // コールスタックを表示
    console.trace("nest1");
    return mFunc(n, 4);
  }
  // 関数その2
  function mFunc(num, n){
    // コールスタックを表示
    console.trace("nest2");
    return Math.pow(num, n);
  }
}, false);
```

ブラウザでindex.htmlを表示すると、デバッグコンソールに呼び出した関数のスタックが表示されます。



ONEPOINT 呼び出し元の関数を調べるには「console.trace()」メソッドを使う

プログラムが複雑化、肥大化すると、どこから呼び出されているのか把握しにくくなります。関数がどこから呼び出されているかを調べるには関数内で「console.trace()」メソッドを使います。「console.trace()」メソッドを使うと、呼び出している関数がスタックで(呼び出し順番に)デバッグコンソールに表示されます。

デバッガによって、どこまで詳しく表示されるか異なりますが、Firebugでは関数に渡されたパラメータの値も確認することができます。

COLUMN 「console.trace()」メソッドの出力結果について

「console.trace()」メソッドの出力結果は、Firefoxの標準のデバッガ以外では、「コンソール」タブを選択すると表示されます。Firefoxの標準のデバッガでは、メニューからWebコンソールを選択すると表示されます。

関連項目 ▶▶▶

- デバッグコンソールに文字を出力する …… p.346
- デバッグコンソールにオブジェクトの内容を出力する …… p.349
- 処理にかかった時間を詳細に把握する …… p.351

CHAPTER 08

DOM

特定のID名を持つ要素にアクセスする

ここでは、特定のID名を持つDOM要素にアクセスする方法について解説します。

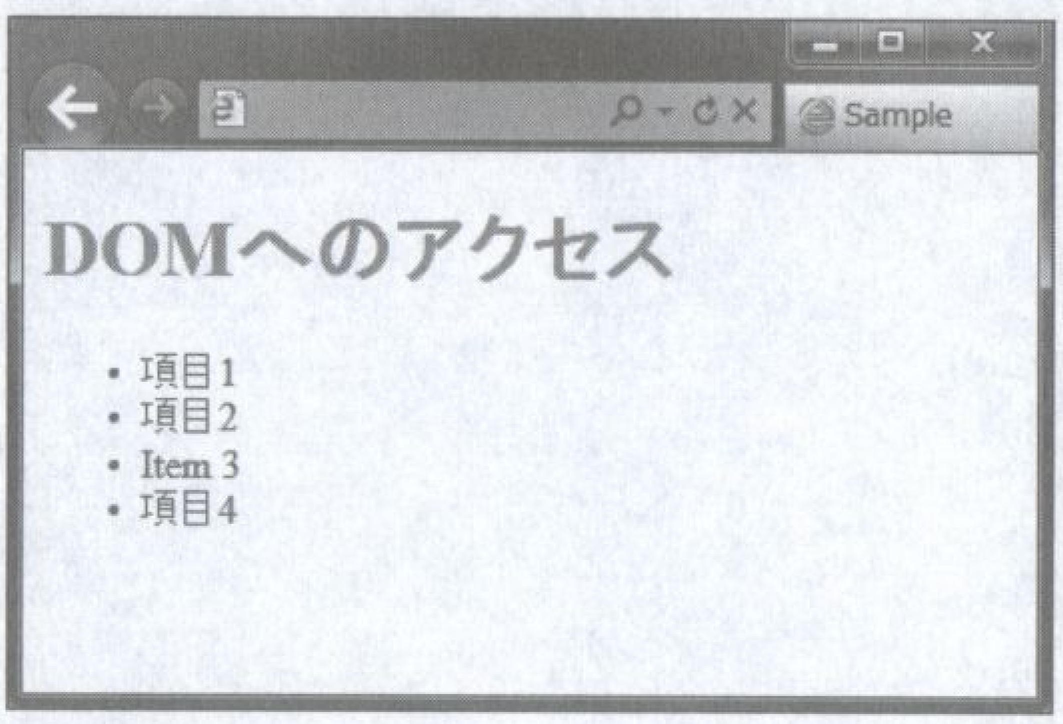
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1 id="myHeader">DOMへのアクセス</h1>
    <ul>
      <li>項目1</li>
      <li>項目2</li>
      <li id="myItem">項目3</li>
      <li>項目4</li>
    </ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 指定したID名を持つ要素の色を赤にする
  document.getElementById("myHeader").style.color = "red";
  // 指定したID名を持つ要素の内容を「Item 3」にする
  document.getElementById("myItem").innerHTML = "Item 3";
}, false);
```

ブラウザでindex.htmlを表示すると、ページ内にある要素に対して操作が行われて、見出し文字の色が赤色に変化し、リスト項目は「項目3」から「Item 3」に内容が変更されます。



ONEPOINT 特定のID名を持つ要素を操作するには「getElementById()」メソッドを使う

ブラウザはページ内の要素は、DOM(Document Object Model:ドキュメントオブジェクトモデル)ツリーとして内部で構築されています。ページ内の要素はDOMツリーとしてアクセスすることができますが、順番にツリーをたどっていくと、アクセスまでに時間がかかります。そこで、要素にID名を指定しておくことでDOMツリーをたどることなく直接、その要素を操作することができます。

直接、要素を操作するには、「getElementById()」メソッドのパラメータに、アクセスしたい要素のID名を文字列で指定します。また、「getElementById()」メソッドで得られた結果を変数に入れておけば、さらに高速に要素に対してアクセスできるようになります。

COLUMN DOMの読み込みタイミング

サンプルでは「load」イベント発生後に要素を操作していますが、現在、使用されている多くのブラウザではDOMの構築ができた際に「DOMContentReady」イベントが発生します。少しでも早くページ内操作を行いたい場合には、「DOMContentReady」イベント発生後に処理を行うのがよいでしょう。また、スマートフォンなどではページの最後に「script」要素を記述することで、より速くページの読み込み処理を行うことができます。また、iOS 5では、「script」要素の「async」属性で、同期か非同期でスクリプトファイルを読み込むという指定ができるので、これらを利用するのもよいでしょう。

関連項目 ▶▶▶

- 特定のスタイルシートクラスを持つ要素にアクセスする ………… p.358
- 要素名を指定してアクセスする ………… p.360
- 特定の名前を持つ要素にアクセスする ………… p.362
- セレクタを使って要素にアクセスする ………… p.364

SECTION-154

特定のスタイルシートクラスを持つ要素にアクセスする

ここでは、特定のスタイルシートクラスを持つ要素にアクセスする方法について解説します。

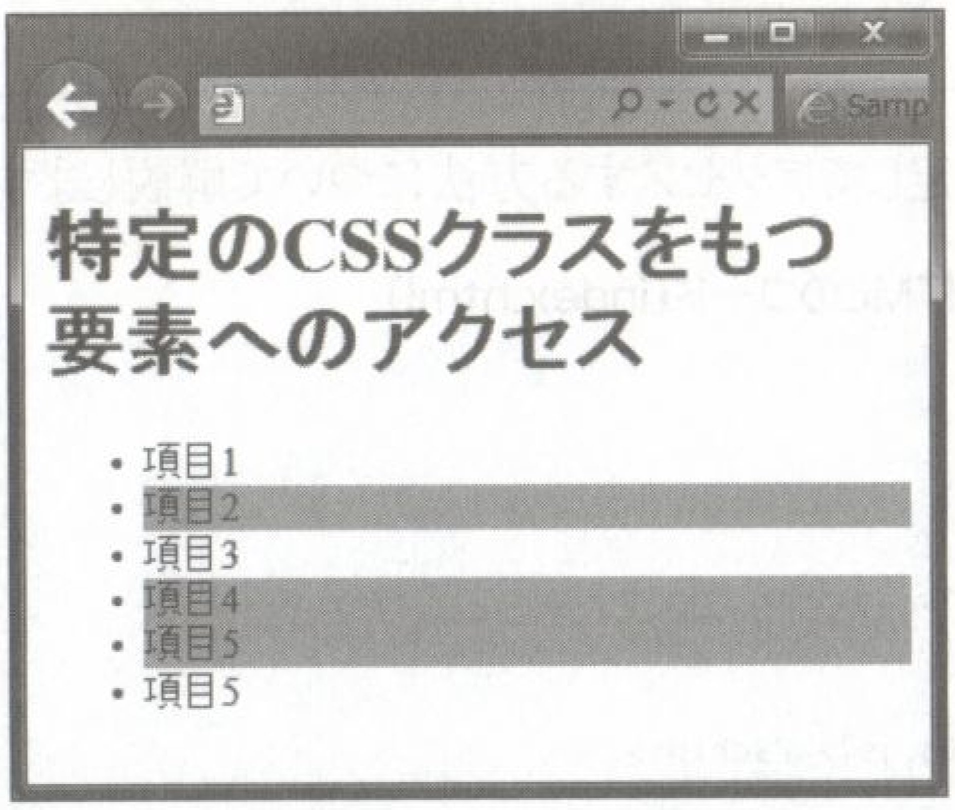
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>特定のCSSクラスをもつ要素へのアクセス</h1>
    <ul>
      <li class="lotus">項目1</li>
      <li class="note">項目2</li>
      <li>項目3</li>
      <li class="note">項目4</li>
      <li class="note">項目5</li>
      <li class="caution">項目5</li>
    </ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 指定したCSSクラスを持つ要素を返す
  var list = document.getElementsByClassName("note");
  // 繰り返し処理し、背景色をオレンジ色にする
  for(var i=0; i<list.length; i++){
    list[i].style.backgroundColor = "orange";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、要素で「note」クラスを持つ要素の背景色がオレンジ色になります。



ONEPOINT 特定のスタイルシートクラスを持つ要素にアクセスするには「getElementsByClassName()」メソッドを使う

要素には、「class」属性でCSSクラスを指定することができます。このCSSクラスで特定の名前が指定された要素を取り出すには、「getElementsByClassName()」メソッドを使います。パラメータには、CSSクラス名を文字列で指定します。

指定できるクラス名が2つ以上ある場合は、「getElementsByClassName("x1 turbo")」のように半角空白で区切って指定します。これは「class」属性に複数のCSSクラスを指定する場合と同じです。

「getElementsByClassName()」メソッドは、該当する要素をノードリストとして返します。JavaScriptでは、ノードリストは配列と同じようにアクセスすることができます。

「getElementsByClassName()」メソッドは、「document」オブジェクトだけでなく、他の要素を基準にして使うこともできます。たとえば、「document.getElementById("main").getElementsByCalssName("mz")」とすると、ID名が「main」の要素内の「mz」クラスを持つ要素だけが対象になります。

関連項目 ▶▶▶

- 特定のID名を持つ要素にアクセスする ……… p.356
- 要素名を指定してアクセスする ……… p.360
- 特定の名前を持つ要素にアクセスする ……… p.362
- セレクタを使って要素にアクセスする ……… p.364

要素名を指定してアクセスする

ここでは、要素名を指定してアクセスする方法について解説します。

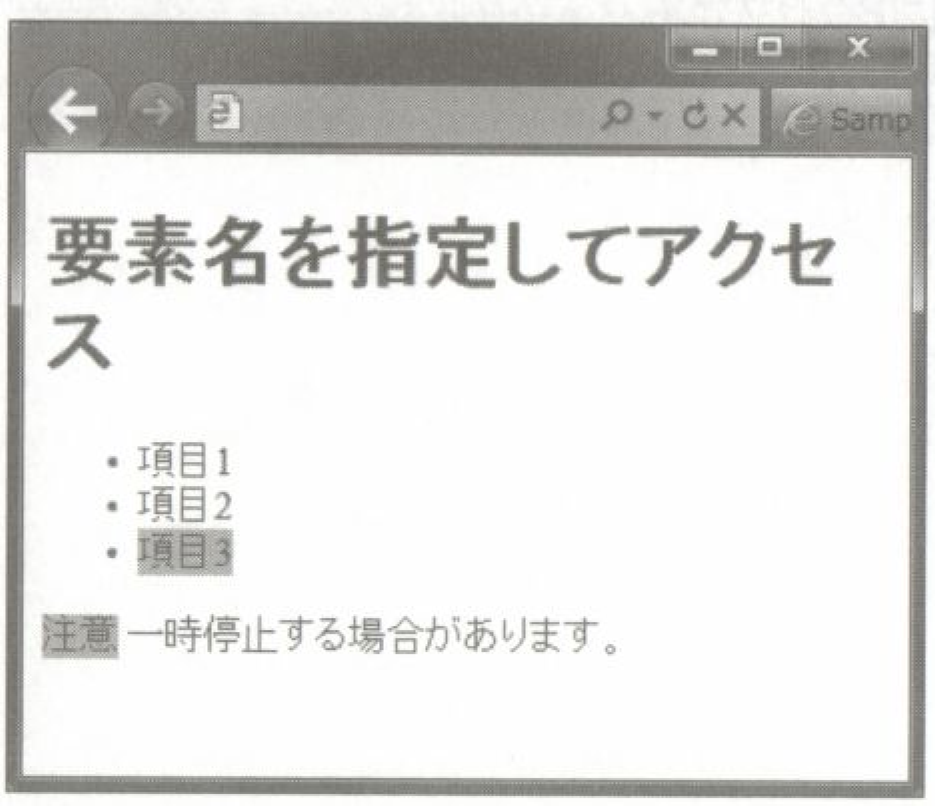
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素名を指定してアクセス</h1>
    <ul>
      <li>項目1</li>
      <li>項目2</li>
      <li><span>項目3</span></li>
    </ul>
    <p><span>注意</span> 一時停止する場合があります。
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 指定した要素名を持つ要素を返す
  var list = document.getElementsByTagName("span");
  // 繰り返し処理し、背景色をオレンジ色にする
  for(var i=0; i<list.length; i++){
    list[i].style.backgroundColor = "orange";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「span」要素の背景色がオレンジ色になります。



ONEPOINT **要素名を指定してアクセスするには「getElementsByTagName()」メソッドを使う**

要素名を指定してアクセスするには、「getElementsByTagName()」メソッドを使います。「getElementsByTagName()」メソッドのパラメータには、要素名を文字列で指定します。

「getElementsByTagName()」メソッドは、「document」オブジェクトだけでなく、他の要素を基準にして使うこともできます。たとえば、「document.getElementById("main").getElementsByTagName("div")」とすると、ID名が「main」の要素内の「div」要素だけが対象になります。

関連項目 ▶▶▶

- 特定のID名を持つ要素にアクセスする …… p.356
- 特定のスタイルシートクラスを持つ要素にアクセスする …… p.358
- 特定の名前を持つ要素にアクセスする …… p.362
- セレクタを使って要素にアクセスする …… p.364

SECTION-156

特定の名前を持つ要素にアクセスする

ここでは、特定の名前を持つ要素にアクセスする方法について解説します。

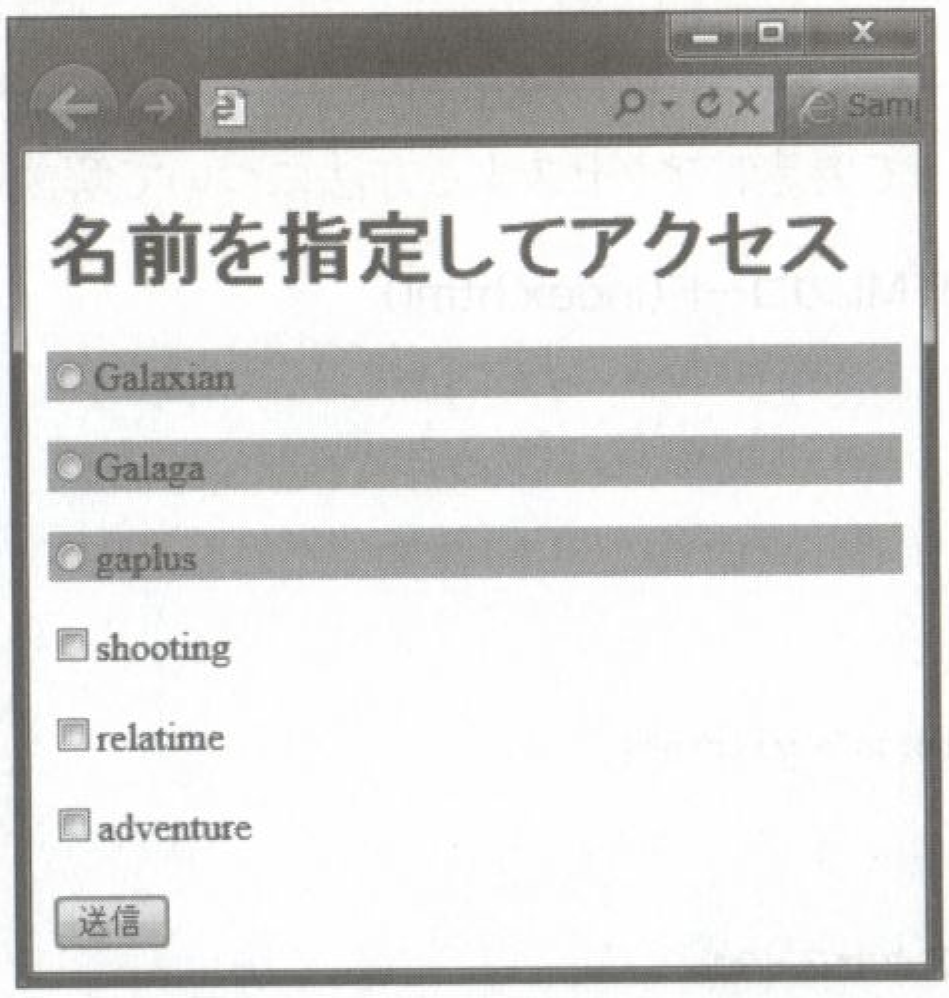
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>名前を指定してアクセス</h1>
    <form>
      <p><input type="radio" name="game">Galaxian</p>
      <p><input type="radio" name="game">Galaga</p>
      <p><input type="radio" name="game">gaplus</p>
      <p><input type="checkbox" name="gtype1">shooting</p>
      <p><input type="checkbox" name="gtype2">relatime</p>
      <p><input type="checkbox" name="gtype3">adventure</p>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 指定した名前を持つ要素を返す
  var list = document.getElementsByName("game");
  // 繰り返し処理し、背景色をオレンジ色にする
  for(var i=0; i<list.length; i++){
    list[i].parentNode.style.backgroundColor = "orange";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「name」属性で「game」の名前を持つラジオボタンの背景色がオレンジ色になります。



ONEPOINT

名前を指定してアクセスするには「getElementsByName()」メソッドを使う

名前を指定してアクセスするには、「getElementsByName()」メソッドを使います。「getElementsByName()」メソッドのパラメータには、名前を文字列で指定します。この名前は、要素の「name」属性で指定した値になります。多くの場合、入力フォーム内のラジオボタンなどへのアクセスに利用することができます。

「getElementsByName()」メソッドは、「document」オブジェクトだけでなく、他の要素を基準にして使うこともできます。たとえば、「document.getElementById("main").getElementsByName("pasopia")」とすると、ID名が「main」の要素内の「name」属性が「pasopia」だけが対象になります。

関連項目 ▶▶▶

- 特定のID名を持つ要素にアクセスする …… p.356
- 特定のスタイルシートクラスを持つ要素にアクセスする …… p.358
- 要素名を指定してアクセスする …… p.360
- セレクタを使って要素にアクセスする …… p.364

SECTION-157

セレクタを使って要素にアクセスする

ここでは、セレクタを使って要素にアクセスする方法について解説します。

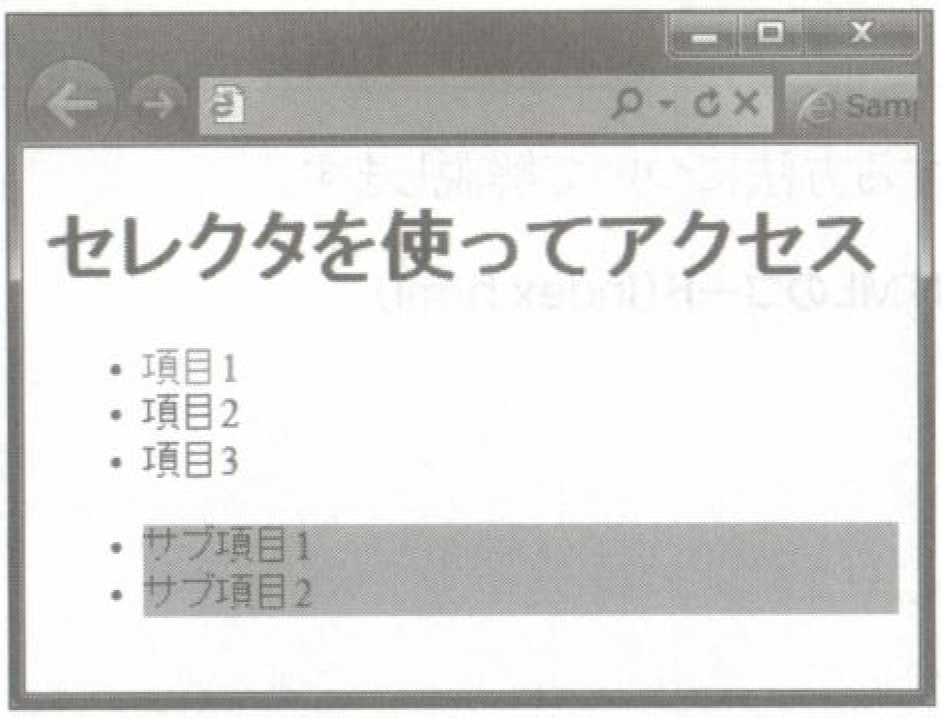
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>セレクタを使ってアクセス</h1>
    <ul id="mz">
      <li><span>項目1</span></li>
      <li>項目2</li>
      <li>項目3</li>
    </ul>
    <div class="note">
      <ul>
        <li>サブ項目1</li>
        <li>サブ項目2</li>
      </ul>
    </div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // li要素にアクセス
  var ele = document.querySelector("#mz li");
  // 最初の文字の色を赤色にする
  ele.firstChild.style.color = "red";
  // noteクラス内のli項目にアクセス
  var list = document.querySelectorAll(".note li")
  // 繰り返し処理し、背景色をオレンジ色にする
  for(var i=0; i<list.length; i++){
    list[i].style.backgroundColor = "orange";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「note」クラス内の「li」項目の背景色がオレンジ色になります。



ONEPOINT

セレクタを使って要素にアクセスするには「querySelector()」「querySelectorAll()」メソッドを使う

セレクタを使って要素にアクセスするには、「querySelector()」メソッド、または、「querySelectorAll()」メソッドを使います。どちらのメソッドもパラメータにCSSセレクタを指定します。

「querySelector()」メソッドは、見つかった最初の要素を返します。該当する要素がない場合は、「null」を返します。

「querySelectorAll()」メソッドは、該当する要素をすべて配列として返します。

「querySelector()」メソッドと「querySelectorAll()」メソッドは、「document」オブジェクトだけでなく、他の要素を基準にして使うこともできます。たとえば、「document.getElementById("main").querySelectorAll(".turbo")」とすると、ID名が「main」の要素内の「turbo」クラスがすべてピックアップされます。

関連項目 ▶▶▶

● 特定のID名を持つ要素にアクセスする …… p.356
● 特定のスタイルシートクラスを持つ要素にアクセスする …… p.358
● 要素名を指定してアクセスする …… p.360
● 特定の名前を持つ要素にアクセスする …… p.362

要素を生成する

ここでは、要素を生成する方法について解説します。

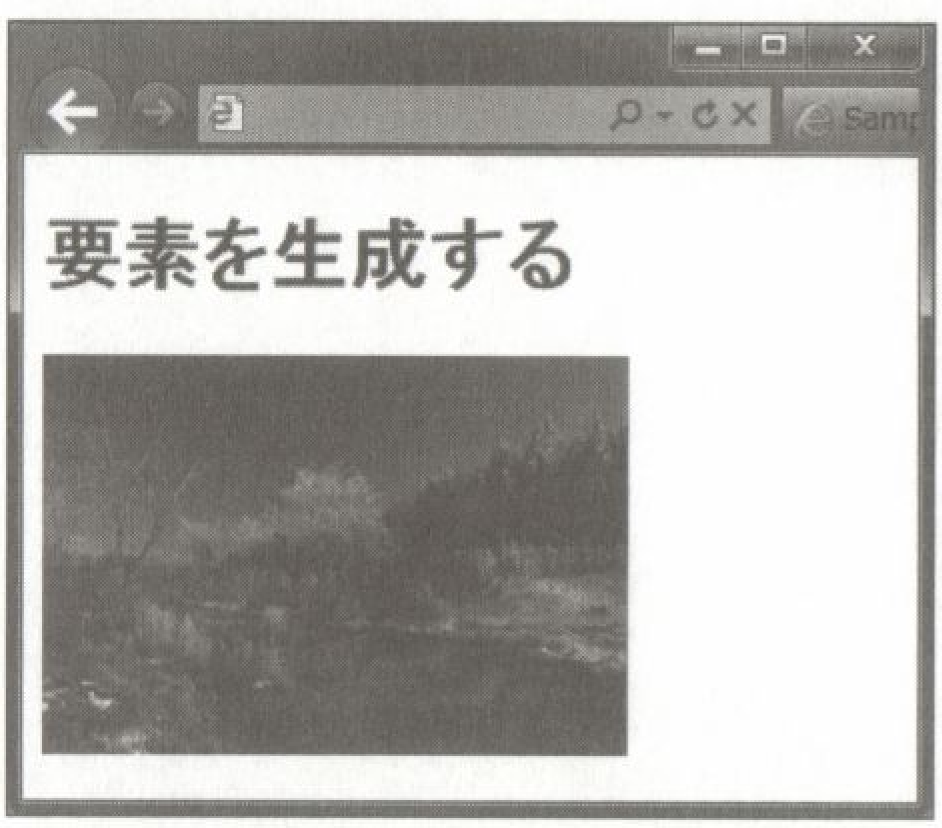
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素を生成する</h1>
    <div></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // img要素を生成
  var myImage = document.createElement("img");
  // 属性(プロパティ)を設定
  myImage.src = "images/sample.jpg";
  myImage.width = 240;
  myImage.height = 160;
  // div要素の末尾に追加
  document.getElementsByTagName("div")[0].appendChild(myImage);
}, false);
```

ブラウザでindex.htmlを表示すると、「img」要素が生成され、「div」要素の末尾に追加されます。また、追加後にページ上に画像が表示されます。



ONEPOINT　要素を生成するには「document.createElement()」メソッドを使う

DOMで要素を生成するには、「document.createElement()」メソッドを使います。「document.createElement()」のパラメータには、要素名を指定します。要素名は大文字、小文字を問いません。

要素を生成しただけではページ上には表示されません。「appendChild()」メソッドなどを使って、ページ上のDOM要素に追加する必要があります。

関連項目 ▶▶▶

- テキストノードを生成する ………… p.368
- ノードを追加・挿入する ………… p.374

SECTION-159

テキストノードを生成する

ここでは、テキストノードを生成する方法について解説します。

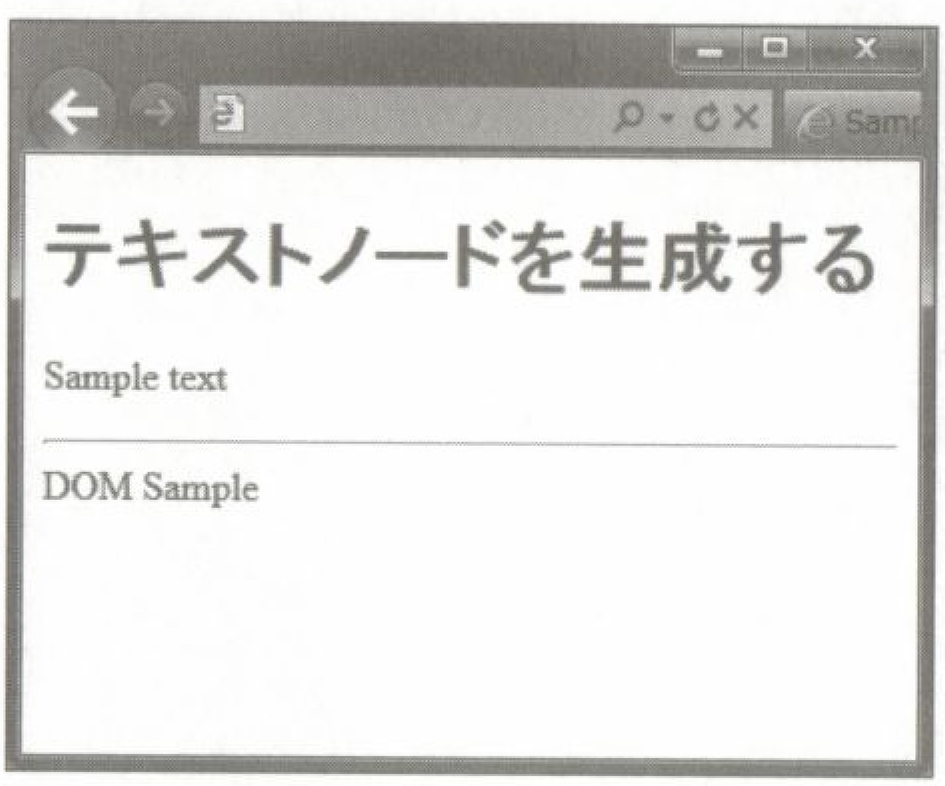
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>テキストノードを生成する</h1>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // テキストノードを生成
  var text1 = document.createTextNode("DOM Sample");
  var text2 = document.createTextNode("Sample text");
  // ページの末尾に追加
  document.body.appendChild(text1);
  // p要素の末尾に追加
  document.getElementsByTagName("p")[0].appendChild(text2);
}, false);
```

ブラウザでindex.htmlを表示すると、テキストノードが生成され、それぞれ、「body」要素の末尾と「p」要素の末尾に追加されます。



ONEPOINT **テキストノードを生成するには「document.createTextNode()」メソッドを使う**

DOMでテキストノードを生成するには、「document.createTextNode()」メソッドを使います。「document.createTextNode()」のパラメータには文字列を指定します。

テキストノードを生成しただけではページ上には表示されません。ノードとして生成されるので、「appendChild()」メソッドなどを使って、ページ上にあるDOM要素に追加や挿入する必要があります。

関連項目 ▶▶▶

- 要素を生成する …… p.366
- ノードを追加・挿入する …… p.374

SECTION-160

子ノードにアクセスする

ここでは、子ノードにアクセスする方法について解説します。

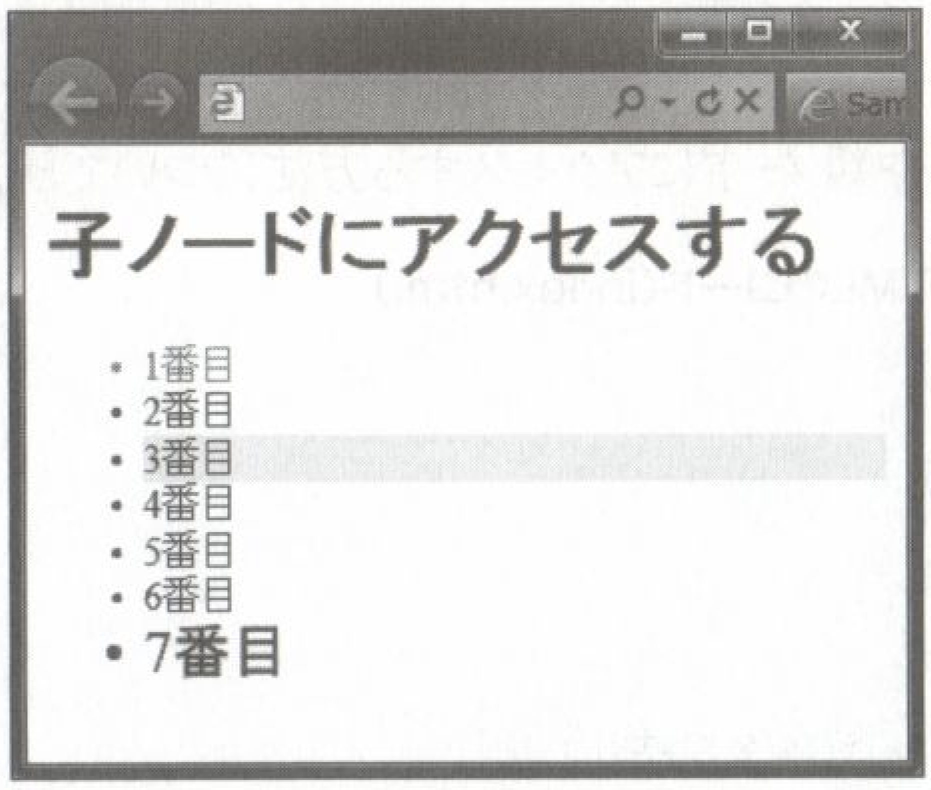
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>子ノードにアクセスする</h1>
    <ul><li>1番目</li>
      <li>2番目</li>
      <li>3番目</li>
      <li>4番目</li>
      <li>5番目</li>
      <li>6番目</li>
      <li>7番目</li></ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ul要素を取得
  var myList = document.getElementsByTagName("ul")[0];
  // 最初のノードの文字色を変更
  myList.firstChild.style.color = "red";
  // 3番目のノードの背景色を変更
  myList.childNodes[4].style.backgroundColor = "yellow";
  // 最後のノードのスタイルを変更
  myList.lastChild.style.fontSize = "18pt";
}, false);
```

ブラウザでindex.htmlを表示すると、「ul」要素の子ノードにアクセスし、それぞれの項目のスタイルが変更されます。



ONEPOINT

子ノードにアクセスするには「firstChild」「childNodes」「lastChild」プロパティを使う

子ノードにアクセスするには、次のいずれかのプロパティを使います。

プロパティ/ノードリスト(配列)	説明
firstChild	最初の子ノード
childNodes[順番]	指定した順番の子ノード(最初のノードは0番目)
lastChild	最後の子ノード

ノードの順番は、ブラウザによって異なります。IE6～8ではHTML要素内に空白などがあると、IE9以降や他のブラウザとは異なった結果になってしまうことがあります。

関連項目 ▶▶▶

●前後のノードや親ノードにアクセスする …… p.372
●子ノードを削除・置換する …… p.376

SECTION-161

前後のノードや親ノードにアクセスする

ここでは、前後のノードや親ノードにアクセスする方法について解説します。

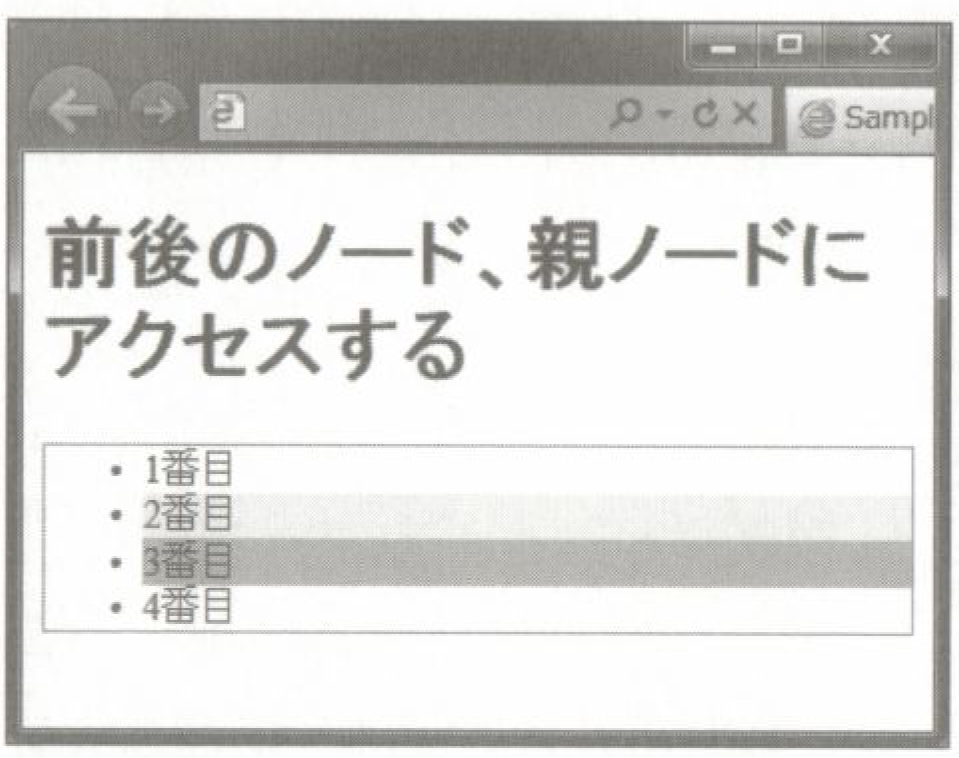
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>前後のノード、親ノードにアクセスする</h1>
    <ul><li>1番目</li><li>2番目</li><li>3番目</li><li>4番目</li></ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ul要素を取得
  var myList = document.getElementsByTagName("ul")[0];
  // 最初の子ノードの次のノードの背景色を変更
  myList.firstChild.nextSibling.style.backgroundColor = "yellow";
  // 最後の子ノードの前のノードの背景色を変更
  myList.lastChild.previousSibling.style.backgroundColor = "orange";
  // 親ノードにアクセス
  myList.firstChild.parentNode.style.border = "1px solid red";
}, false);
```

ブラウザでindex.htmlを表示すると、最初のノードの後ろの項目の背景色と最後のノードの前の項目の背景色が変更され、さらに親ノードである「ul」要素に赤枠が表示されます。



ONEPOINT 前後のノードにアクセスするには「nextSibling」「previousSibling」プロパティ、親ノードにアクセスするには「parentNode」プロパティを使う

前後のノード、および、親ノードにアクセスするには、次のプロパティを使います。

プロパティ	説明
nextSibling	次のノード
previousSibling	前のノード
parentNode	親ノード

ノードの順番は、ブラウザによって異なります。IE6~8ではHTML要素内に空白などがあると、IE9以降や他のブラウザとは異なった結果になってしまうことがあります。次のURLを参照してください。

URL http://blogs.msdn.com/b/ie_jp/archive/2011/10/03/10219055.aspx

関連項目 ▶▶▶

● 子ノードにアクセスする ………………………………………………… p.370

SECTION-162

ノードを追加・挿入する

DOMには要素の末尾に追加するメソッドと、要素の前に挿入するメソッドが用意されています。ここでは、ノードを追加・挿入する方法について解説します。

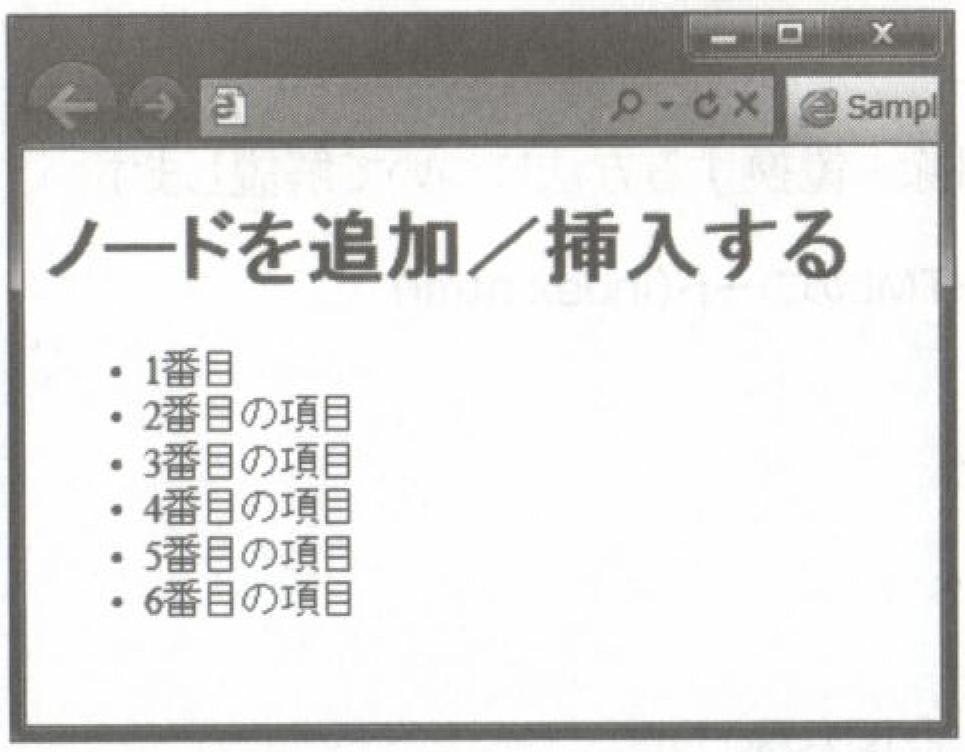
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ノードを追加／挿入する</h1>
    <ul><li>1番目</li></ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ul要素を取得
  var myList = document.getElementsByTagName("ul")[0];
  // リスト項目を5つ追加
  for(var i=2; i<7; i++){
    (function(){
      // 新規にli要素を作成
      var myItem = document.createElement("li");
      // テキストノードを追加
      var myText = document.createTextNode(i+"番目の項目");
      // li要素に作成したテキストノードを追加
      myItem.appendChild(myText);
      // 子ノードとしてli要素を追加
      myList.appendChild(myItem);
    })();
  }
}, false);
```

ブラウザでindex.htmlを表示すると、「li」要素と、その子ノードである文字が追加されます。追加された項目には、「～の項目」という文字が付加されています。



ONEPOINT **ノードを追加するには「appendChild()」メソッド、挿入するには「insertBefore()」メソッドを使う**

ノードを追加・挿入するには、次のメソッドを使います。

メソッド	説明
appendChild(node)	子ノードを追加する
insertBefore(node, pos)	基準ノードの前に挿入する

ノードの順番は、ブラウザによって異なります。IE6～8ではHTML要素内に空白などがあると、IE9以降や他のブラウザとは異なった結果になってしまうことがあります。

関連項目 ▶▶▶

● 要素を生成する …… p.366
● テキストノードを生成する …… p.368
● 子ノードを削除・置換する …… p.376

SECTION-163

子ノードを削除・置換する

ここでは、子ノードを削除・置換する方法について解説します。

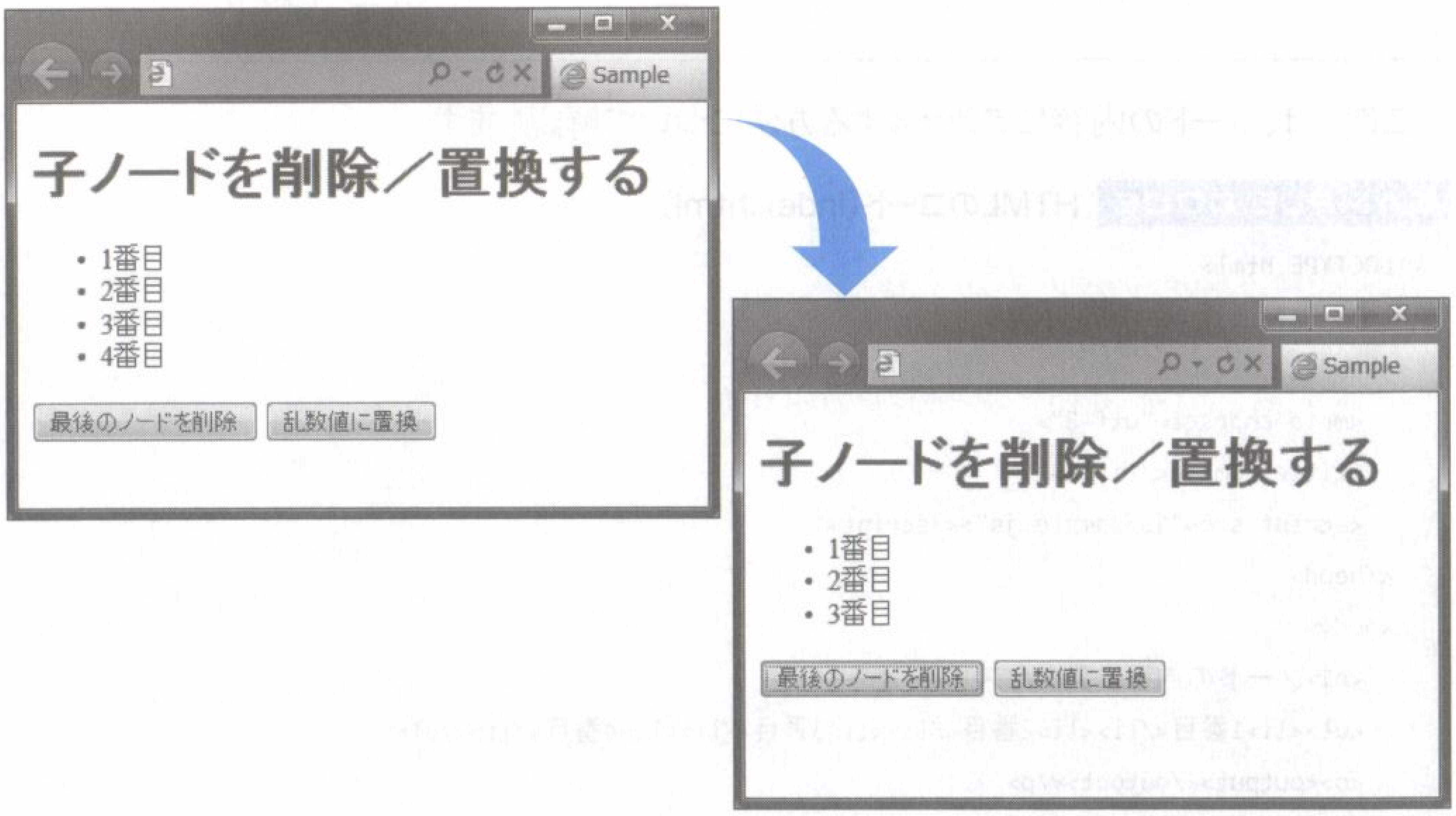
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>子ノードを削除/置換する</h1>
    <ul><li>1番目</li><li>2番目</li><li>3番目</li><li>4番目</li></ul>
    <form>
      <input type="button" id="delBtn" value="最後のノードを削除">
      <input type="button" id="repBtn" value="乱数値に置換">
    </form>
  </body>
</html>
```

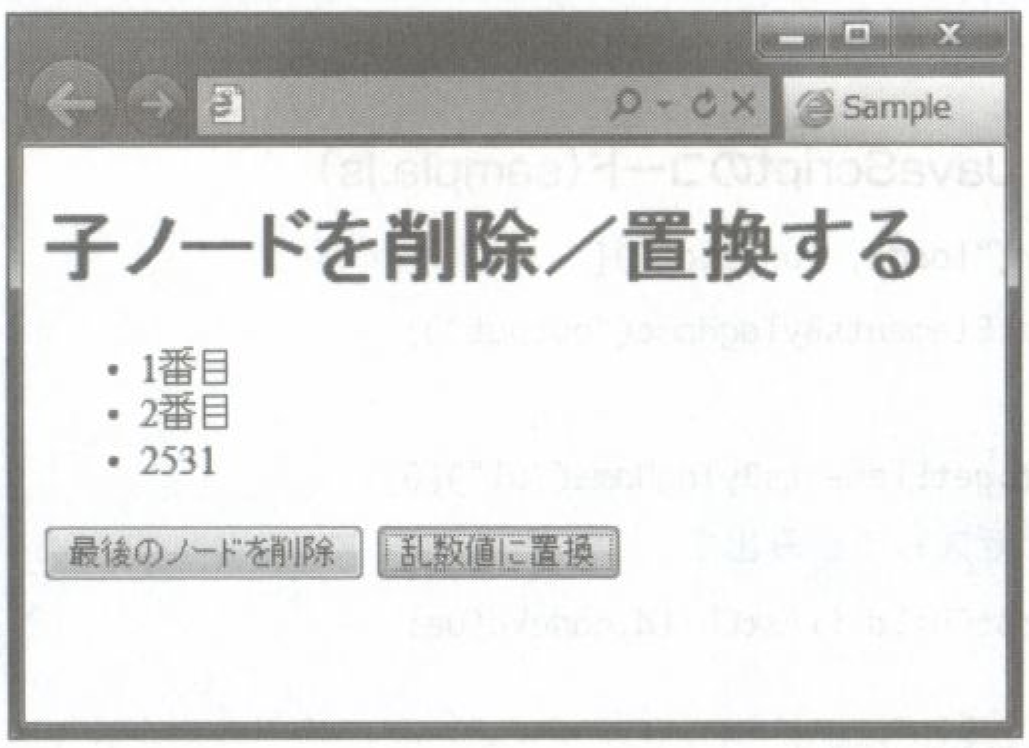
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 削除ボタンにイベントを割り当て
  document.getElementById("delBtn").addEventListener("click", function(){
    // ul要素を取得
    var myList = document.getElementsByTagName("ul")[0];
    // 最後のノードを削除。削除したノードを変数に代入(使用はしない)
    var oldChild = myList.removeChild(myList.lastChild);
  }, false);
  // 置換ボタンにイベントを割り当て
  document.getElementById("repBtn").addEventListener("click", function(){
    // ul要素を取得
    var myList = document.getElementsByTagName("ul")[0];
    // 整数の乱数値を求める(32bit範囲)
    var text = document.createTextNode((Math.random() * 10000) | 0);
    // li要素を作成
    var item = document.createElement("li");
    item.appendChild(text);
    // 最後のノードを置換。置換前のノードを変数に代入(使用はしない)
    var repNode = myList.replaceChild(item, myList.lastChild);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「最後のノードを削除」ボタンをクリックすると、最後の「li」項目が削除されます。



また、「乱数値に置換」ボタンをクリックすると、最後の項目のテキストが乱数値に置換されます。



ONEPOINT　子ノードを削除するには「removeChild()」メソッド、置換するには「replaceChild()」メソッドを使う

子ノードを削除・置換するには、次のメソッドを使います。

メソッド	説明
removeChild(node)	子ノードを削除する
replaceChild(node, pos)	子ノードを置換する

ノードの順番は、ブラウザによって異なります。IE6〜8ではHTML要素内に空白などがあると、IE9以降や他のブラウザとは異なった結果になってしまうことがあります。

関連項目 ▶▶▶

● 要素を生成する …………………………………………………… p.366
● テキストノードを生成する ……………………………………… p.368
● ノードを追加・挿入する ………………………………………… p.374

ノードの内容にアクセスする

ここでは、ノードの内容にアクセスする方法について解説します。

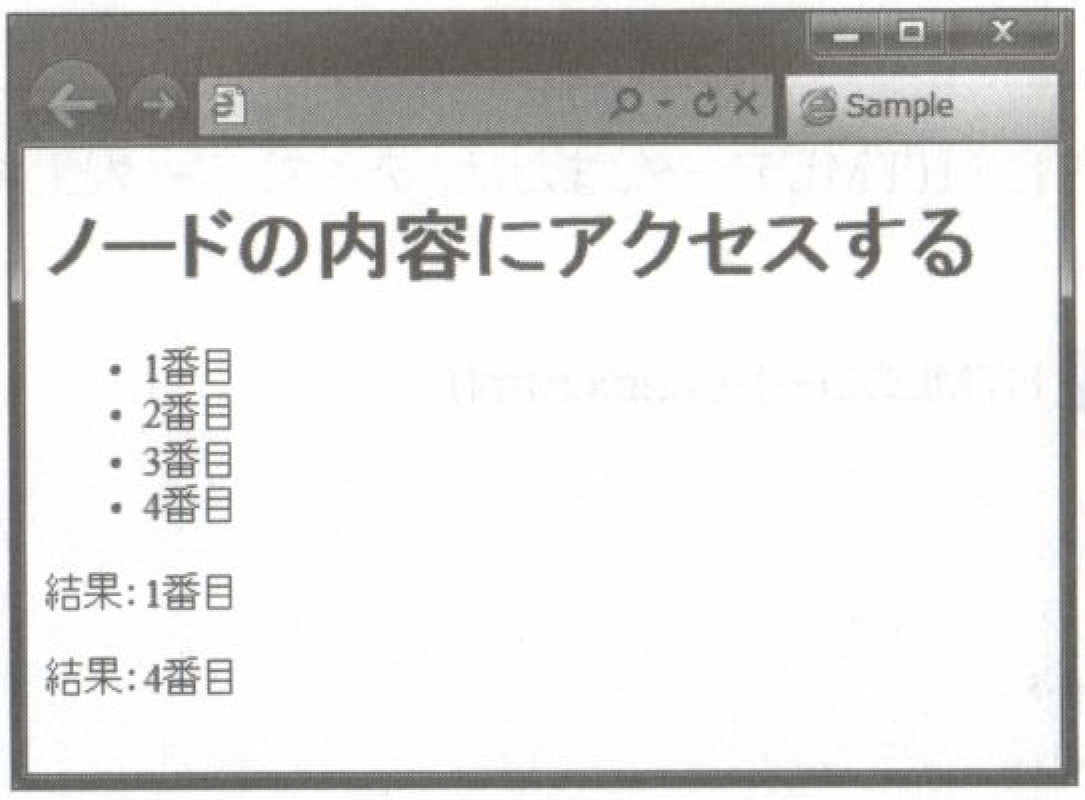
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ノードの内容にアクセスする</h1>
    <ul><li>1番目</li><li>2番目</li><li>3番目</li><li>4番目</li></ul>
    <p><output></output></p>
    <p><output></output></p>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // ul要素を取得
  var myList = document.getElementsByTagName("ul")[0];
  // 最初のノードにアクセスして読み出す
  var val1 = myList.firstChild.firstChild.nodeValue;
  // 結果を出力
  ele[0].innerHTML = "結果 :"+val1;
  // 最後のノードにアクセスして読み出す
  var val2 = myList.lastChild.firstChild.nodeValue;
  // 結果を出力
  ele[1].innerHTML = "結果 :"+val2;
}, false);
```

ブラウザでindex.htmlを表示すると、最初と最後のノードの内容を読み出した結果が表示されます。



ONEPOINT　ノードの内容にアクセスするには「nodeValue」プロパティを参照する

ノードの内容にアクセスするには、「nodeValue」プロパティを参照します。「nodeValue」プロパティは、読み出す要素によって内容が異なります。

属性	属性値
CDATA	CDATAセクションの内容
コメント	コメント内容
ドキュメント	nullになる
DocumentDragment	nullになる
DocumentType	nullになる
要素/エレメント	nullになる
ノードマップ	nullになる
実体参照	nullになる
ノーテーション	nullになる
処理命令	内容
テキスト	文字

SECTION-165

要素の内容を書き換える

ここでは、要素の内容をHTMLデータ、または、テキストデータとして書き換える方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>内容を書き換える</h1>
    <form>
      <textarea cols="40" rows="10">Sample <b>text</b> です。
      </textarea><hr>
      <input type="button" id="htmlBtn" value="innerHTMLで書き換え">
      <input type="button" id="textBtn" value="innerText/textContentで書き換え">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // innerHTMLで書き換えボタンにイベントを割り当て
  document.getElementById("htmlBtn").addEventListener("click", function(){
    // textarea要素の内容を読み出し
    var text = document.getElementsByTagName("textarea")[0].value;
    // ページ上に出力
    document.getElementsByTagName("output")[0].innerHTML = text;
  }, false);
  // innerTextで書き換えボタンにイベントを割り当て
  document.getElementById("textBtn").addEventListener("click", function(){
    // textarea要素の内容を読み出し
    var text = document.getElementsByTagName("textarea")[0].value;
    // ページ上に出力。Firefoxの場合
    var ele = document.getElementsByTagName("output")[0];
    ele.textContent = text+"(Firefox)";
    // Firefox以外の場合
    ele.innerText = text;
  }, false);
```

```
}, false);
```

ブラウザでindex.htmlを表示し、「innerHTMLで書き換え」ボタンをクリックすると、テキストエリアの内容をHTMLタグを解釈して表示されます。



また、「innerText/textContentで書き換え」ボタンをクリックすると、テキストエリアの内容をHTMLタグを解釈せずに表示されます。



ONEPOINT

要素の内容を書き換えるには「innerHTML」プロパティや「innerText」プロパティを使う

要素の内容を書き換えるには、2つの方法があります。1つはHTMLとして解釈させる場合で、「innerHTML」プロパティにアクセスします。もう1つはプレーンテキスト(標準テキスト)として解釈させる場合で、「innerText」プロパティにアクセスします。ただし、Firefoxでは、「innerText」プロパティの代わりに、「textContent」プロパティにアクセスします。

なお、「innerHTML」プロパティや「innerText」プロパティ、「textContent」プロパティは書き換えだけでなく、要素の内容の読み出しを行うこともできます。

SECTION-166

属性値を取得・設定する

ここでは、属性値を取得・設定する方法について解説します。

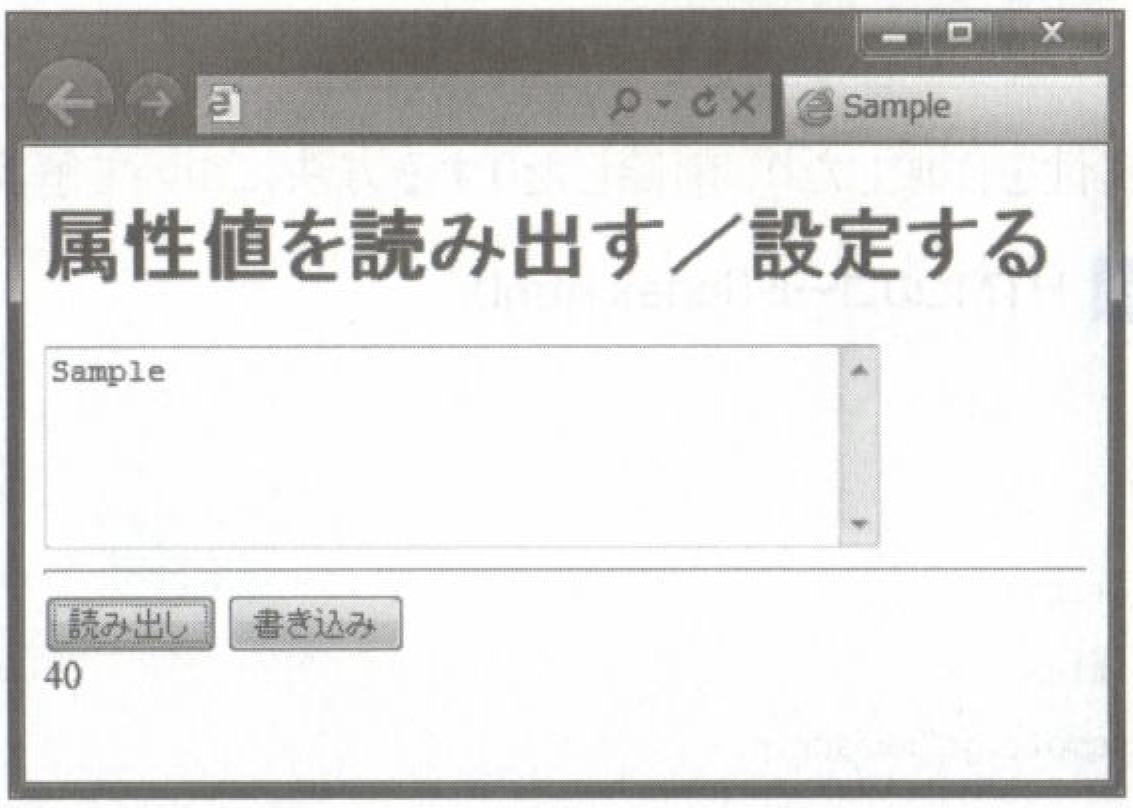
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>属性値を読み出す／設定する</h1>
    <form>
      <textarea cols="40" rows="5">Sample</textarea><hr>
      <input type="button" id="getBtn" value="読み出し">
      <input type="button" id="setBtn" value="書き込み">
    </form>
    <output></output>
  </body>
</html>
```

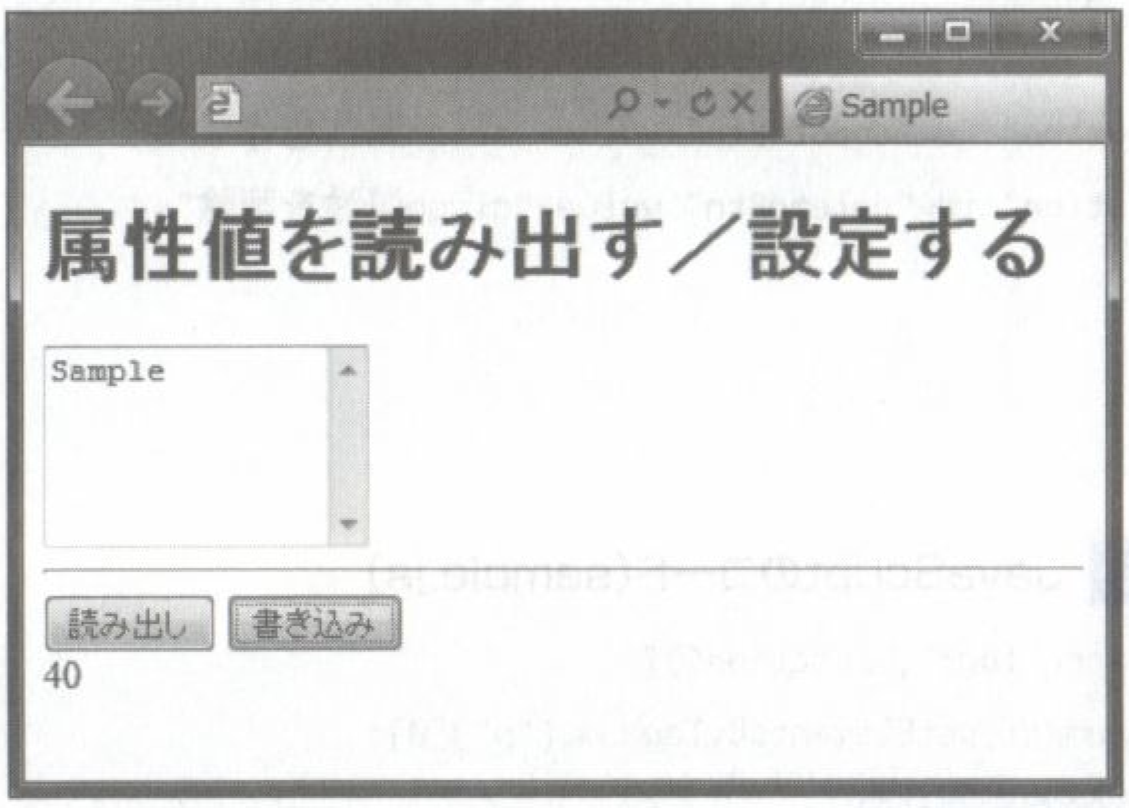
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // textarea要素への参照を変数に代入
  var te = document.getElementsByTagName("textarea")[0];
  // 読み出しボタンにイベントを割り当て
  document.getElementById("getBtn").addEventListener("click", function(){
    // textarea要素のcols属性を読み出し
    var val = te.getAttribute("cols");
    // ページ上に出力
    ele.innerHTML = val;
  }, false);
  // 書き込みボタンにイベントを割り当て
  document.getElementById("setBtn").addEventListener("click", function(){
    // textarea要素のcols属性に値を設定
    var size = (Math.random() * 20) | 0 + 10;
    te.setAttribute("cols", size);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「読み出し」ボタンをクリックすると、テキストエリアの「cols」属性の値を読み出した結果が表示されます。



また、「書き込み」ボタンをクリックすると、テキストエリアの「cols」属性にランダムな値が書き込まれ、テキストエリアの横幅が変わります。



ONEPOINT **属性値を取得するには「getAttribute()」メソッド、設定するには「setAttribute()」メソッドを使う**

要素の属性値を取得する（読み出す）には、「getAttribute()」メソッドを使います。パラメータには取得したい属性名を指定します。

属性に値を設定する場合は、「setAttribute()」メソッドを使います。最初のパラメータに属性名を文字列で指定し、2番目のパラメータに設定値を指定します。

COLUMN **IE6/IE7での「class」属性**

IE6/IE7では、「getAttribute("class")」としても「class」属性の値を読み出すことができません。IE6/IE7では、「getAttiribute("className")」とする必要があります。

SECTION-167

属性を新規作成・削除する

ここでは、新規に属性を作成したり、削除したりする方法について解説します。

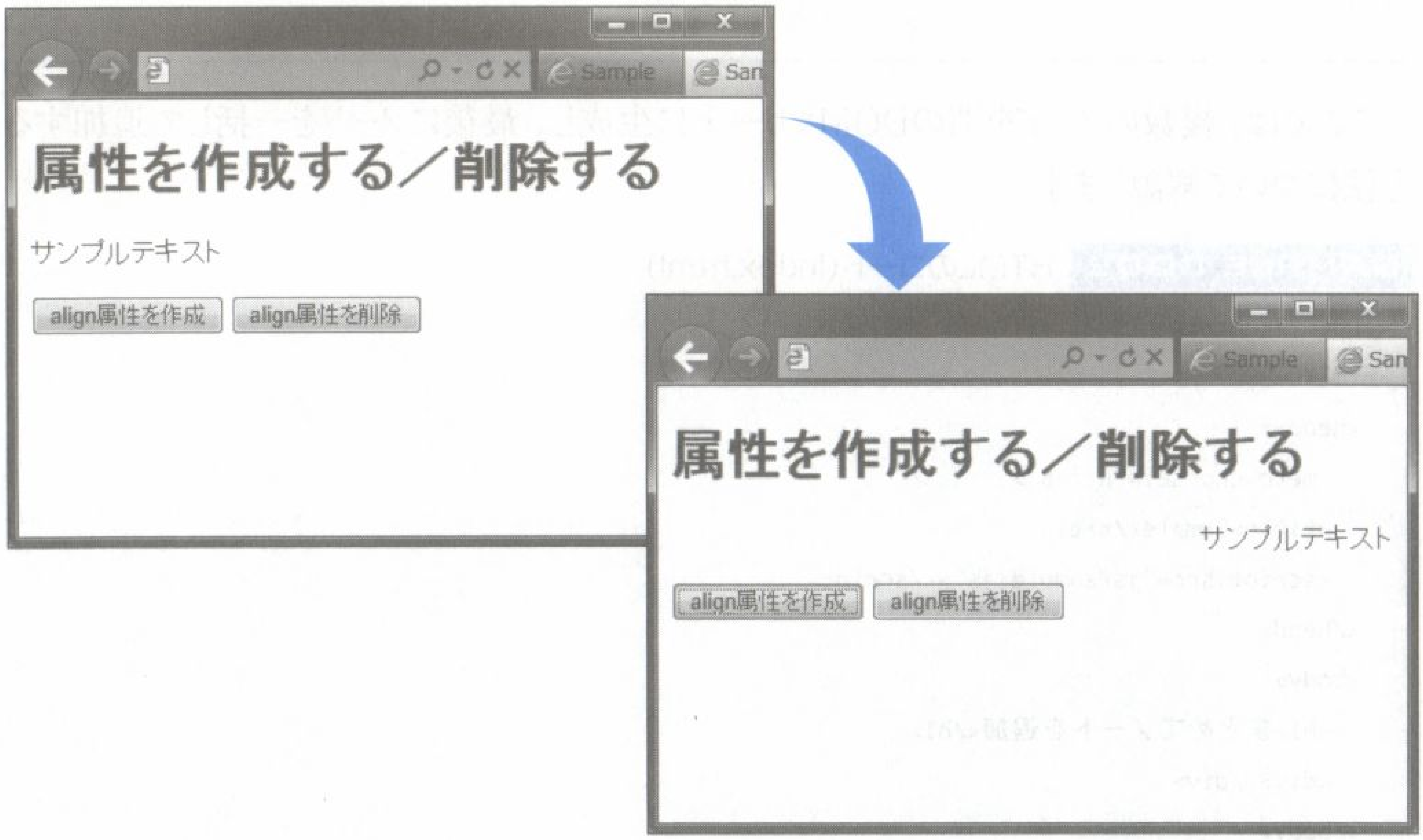
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>属性を作成する／削除する</h1>
    <p>サンプルテキスト</p>
    <form>
      <input type="button" id="createBtn" value="align属性を作成">
      <input type="button" id="deleteBtn" value="align属性を削除">
    </form>
  </body>
</html>
```

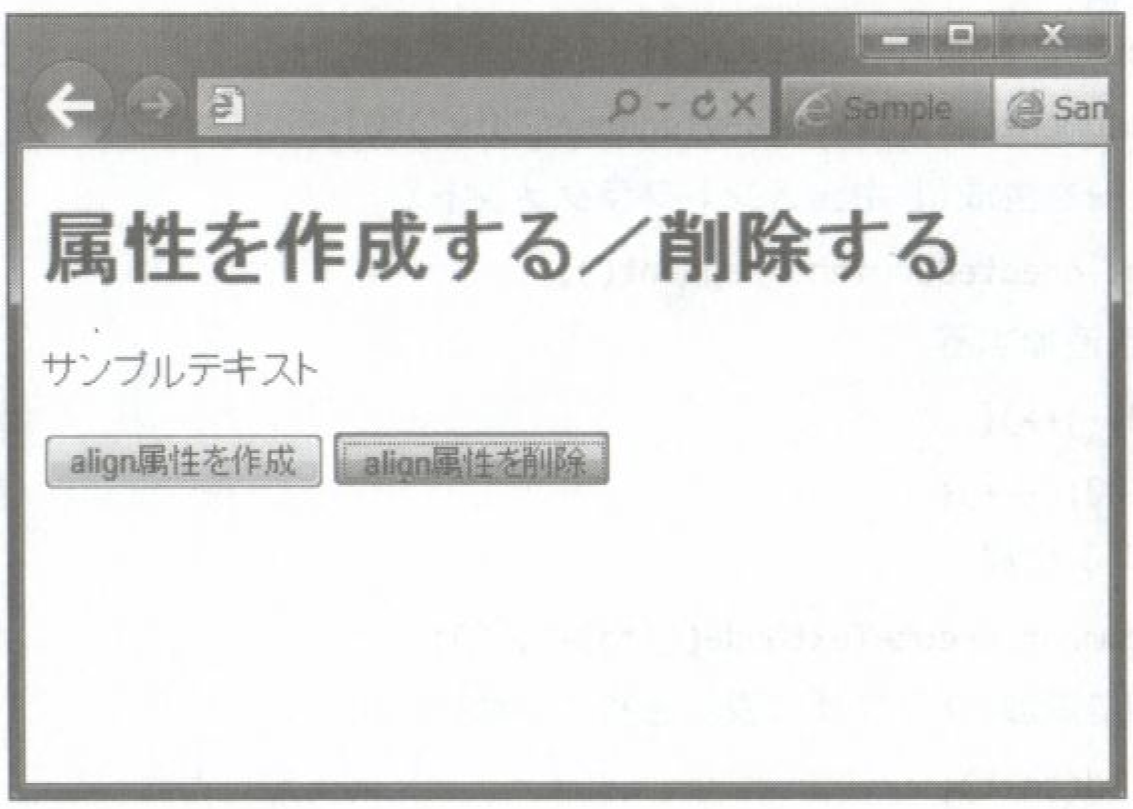
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var paragraph = document.getElementsByTagName("p")[0];
  // 作成ボタンにイベントを割り当て
  document.getElementById("createBtn").addEventListener("click", function(){
    // align属性を作成
    var attr = document.createAttribute("align");
    // align属性にrightを割り当て
    attr.nodeValue = "right";
    // 段落に属性を設定
    paragraph.setAttributeNode(attr);
  }, false);
  // 削除ボタンにイベントを割り当て
  document.getElementById("deleteBtn").addEventListener("click", function(){
    // talign属性を削除
    paragraph.removeAttribute("align");
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「align属性を作成」ボタンをクリックすると、「p」要素に「align」属性が設定、値が「right」になり、「p」要素内の文字は右揃えになります。



また、「align属性を削除」ボタンをクリックすると「p」要素の「align」属性が削除され、「p」要素内の文字はデフォルトの左揃えになります。



ONEPOINT　新規に属性を作成するには「document.createAttribute()」メソッド、削除するには「removeAttribute()」メソッドを使う

新規に属性を作成するには、「document.createAttribute()」メソッドを使います。パラメータには作成する属性名を文字列で指定します。「document.createAttribute()」は作成した属性をオブジェクトとして返します。属性に値を設定するには、「nodeValue」プロパティにアクセスします。

属性を削除するには、対象となる要素の「removeAttribute()」メソッドを使います。パラメータには削除する属性名を文字列で指定します。

SECTION-168

ノードを一括して追加する

ここでは、複数のノードを別のDOMツリー上に生成し、最後にノードを一括して追加する方法について解説します。

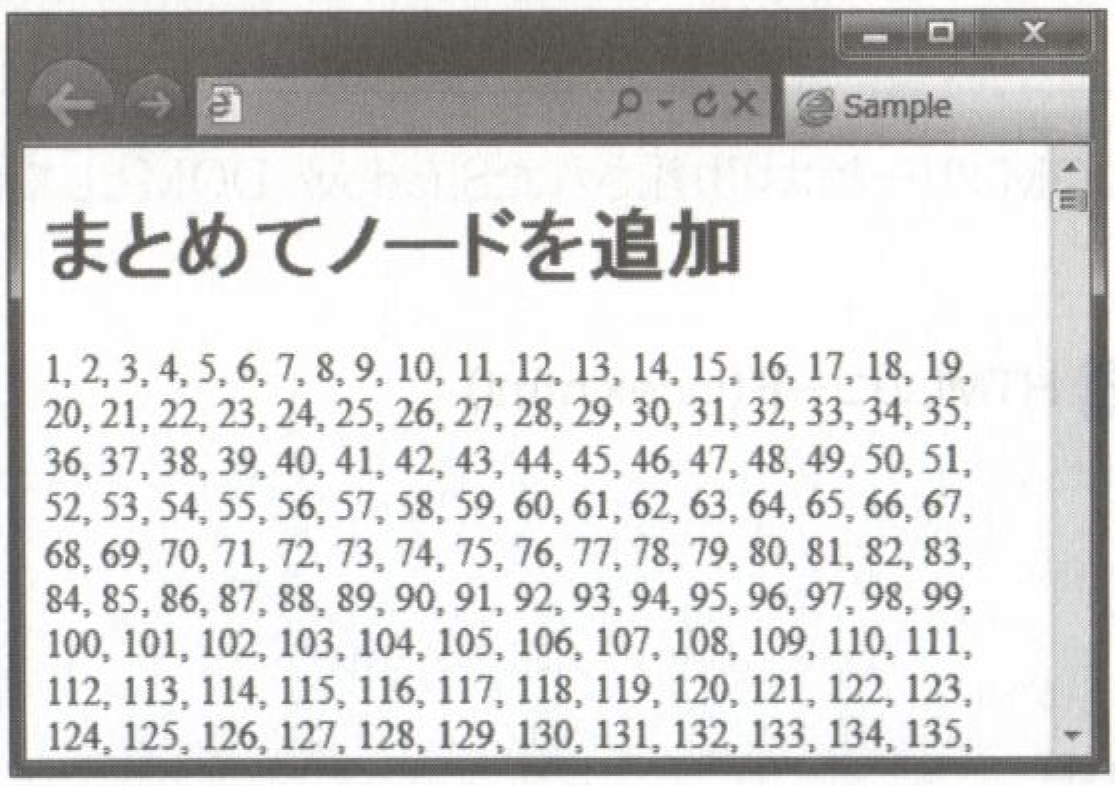
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>まとめてノードを追加</h1>
    <div></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var paragraph = document.getElementsByTagName("div")[0];
  // 別領域のDOMツリーを生成(ドキュメントフラグメント)
  var docf = document.createDocumentFragment();
  // 繰り返しノードを追加する
  for(var j=1; j<100; j++){
    for(var i=1; i<400; i++){
      // テキストノード生成
      var text = document.createTextNode((i*j)+", ");
      // 別領域のDOMに追加(ブラウザに表示されていないDOM)
      docf.appendChild(text);
    }
    // 改行コードを追加
    docf.appendChild(document.createElement("br"));
  }
  // 最後にブラウザ上のドキュメントに表示
  paragraph.appendChild(docf);
}, false);
```

ブラウザでindex.htmlを表示すると、別のDOMツリーにノードを生成し、最後にまとめてページ上に追加されます。



ONEPOINT **ノードを一括して追加するには「document.createDocumentFragment()」メソッドを使って他のDOMツリーを用意する**

JavaScriptでは、他のDOMツリーを用意して、ノードを一括して追加することができます。ブラウザ上に表示されているDOMツリーとは別に作成するには、「document.createDocumentFragment()」メソッドを使います。これにより、ブラウザに表示されているDOMツリーとは切り離されたDOMツリーが生成されます。このDOMツリーはページ上に表示されている要素に追加しない限り、表示されることはありません。

「document.createDocumentFragment()」メソッドは生成した新たなDOMツリーを返します。これが(ドキュメント)ルートになります。あとは、DOMを操作するメソッドを使ってノードの追加、削除などを行います。

ページ上に表示するには、ページ上にある要素に対して「appendChild()」メソッドなどを使って追加します。「appendChild()」メソッドは追加という形になっていますが、「document.createDocumentFragment()」メソッドによって生成したノードはブラウザのドキュメント上に移動する形になります。

COLUMN **ノードを追加する際の速度について**

一括してノードを追加した方が速いように思えますが、実際に計測してみると効果が上がらなかったり、かえって速度が低下してしまうことがあります。

SECTION-169

Shadow DOMとして要素を追加する

ここでは、通常のDOMツリーとは切り離されたShadow DOMとして要素を追加する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      b { color: red; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>Shadow DOM</h1>
    <div>通常の<b>DOMツリー</b></div>
    <div id="host"></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Shadow DOMのホストとなる要素を読み出し
  var shadowHost = document.getElementById("host");
  // Shadow DOMのルートを指定
  var shadowRoot = new WebKitShadowRoot(shadowHost);
  // p要素を作成
  var p = document.createElement("p");
  p.innerHTML = "<b>Test</b> (Shadow DOM)";
  // Shadow DOMに追加
  shadowRoot.appendChild(p);
  // p要素を作成
  p = document.createElement("p");
  p.innerHTML = "<b>Test</b> (通常のDOMツリー)";
  // 通常のDOMツリーに追加
  document.body.appendChild(p);
}, false);
```

ブラウザでindex.htmlを表示し、ページ内にShadow DOMとして段落を追加して、その後、通常のDOMにも段落を追加します。Shadow DOMとして追加した要素は通常のDOMツリーとは切り離されているため、CSSが適用されません。



ONEPOINT Shadow DOMとして追加するには「new WebKitShadowRoot()」とする

Shadow DOMはページ上には表示されますが、通常のDOMツリーとは切り離されています。このため、スタイルシートが適用されません。また、通常のDOMツリーでのアクセスからも切り離されています(カプセル化されている)。

このShadow DOMとして要素を追加するには、「new WebKitShadowRoot()」としてShadow DOMのルートを作成します(2012年8月時点ではベンダープレフィックスを付ける必要がある)。パラメータには、Shadow DOMのルートとなる要素を指定します。これは、通常のDOMツリーにある要素を指定することになります。

作成したShadow DOMは、一般的なDOM処理を行うメソッドなどを使って、要素を追加したり、削除したりすることができます。

COLUMN Chromeでの設定

Google Chrome 21では標準でShadow DOMが動作しません。Shadow DOMを動作させるには、アドレスバーに「about:flags」と入力し、表示される設定画面で「Shadow DOMを有効にする」の項目で機能するように設定を行います。

関連項目 ▶▶▶

● 通常のDOMツリーの要素をShadow DOMにする ……………… p.390
● Shadow DOMにある要素にアクセスする ……………… p.392

SECTION-170

通常のDOMツリーの要素をShadow DOMにする

ここでは、通常のDOMツリーの要素をShadow DOMにする方法について解説します。

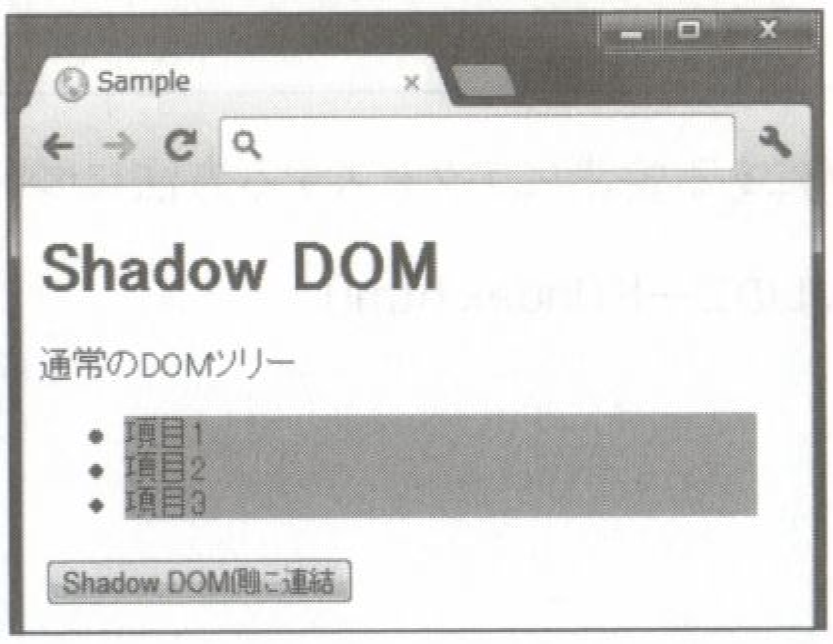
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      li { background-color: orange; width: 300px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>Shadow DOM</h1>
    <div>通常のDOMツリー</div>
    <div id="host"></div>
    <ul id="menu">
      <li>項目1</li>
      <li>項目2</li>
      <li>項目3</li>
    </ul>
    <form>
      <input type="button" value="Shadow DOM側に連結" id="conect">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  document.getElementById("conect").addEventListener("click", function(){
    // Shadow DOMのホストとなる要素を読み出し
    var shadowHost = document.getElementById("host");
    // Shadow DOMのルートを指定
    var shadowRoot = new WebKitShadowRoot(shadowHost);
    // Shadow DOMに連結するリスト項目
    var myList = document.getElementById("menu");
    // Shadow DOMに追加
    shadowRoot.appendChild(myList);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると。リスト項目は通常のDOMツリーとして存在しています。このため、CSSで設定した背景色が反映されています。



ボタンをクリックすると、通常のDOMツリーからShadow DOM側に連結されます。これにより、通常のCSSは反映されなくなり、背景色はデフォルト(透明)になります。



ONEPOINT 要素をShadow DOMにするには「new WebKitShadowRoot()」とする

通常のDOMツリーに存在する要素をShadow DOMにするには、「new WebKitShadowRoot()」としてShadow DOMのルート要素を生成します。パラメータには、ホストとなる通常のDOM要素を指定します。

次にShadow DOMのルートの「appendChild()」メソッドを使って、Shadow DOMにしたい要素をパラメータとして指定します。これで、通常のDOMからShadow DOMに要素が移動します。この時点でCSSなどは通常のDOMから切り離され、反映されなくなります。JavaScriptでの通常のDOM要素のアクセスからも切り離されます。

COLUMN Chromeでの設定

Google Chrome 21では標準でShadow DOMが動作しません。Shadow DOMを動作させるには、アドレスバーに「about:flags」と入力し、表示される設定画面で「Shadow DOMを有効にする」の項目で機能するように設定を行います。

関連項目 ▶▶▶

- Shadow DOMとして要素を追加する ……… p.388
- Shadow DOMにある要素にアクセスする ……… p.392

SECTION-171

Shadow DOMにある要素にアクセスする

ここでは、Shadow DOMにある要素にアクセスする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

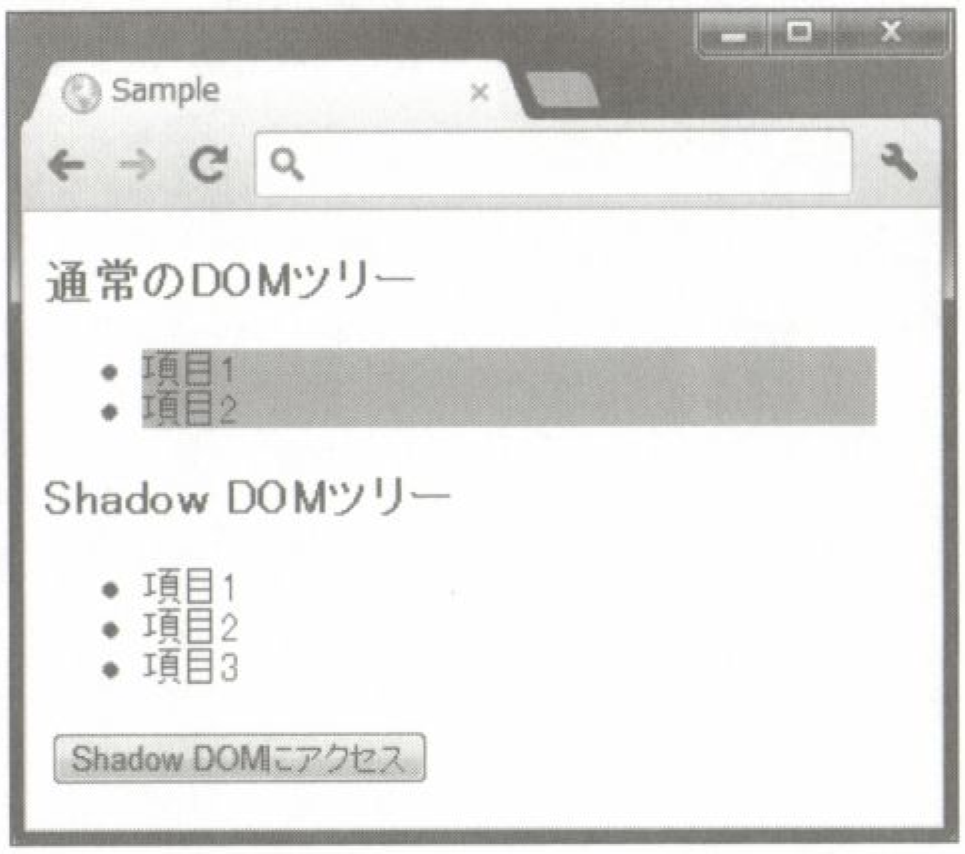
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      li { background-color: orange; width: 300px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>通常のDOMツリー</h3>
    <ul>
      <li>項目1</li>
      <li>項目2</li>
    </ul>
    <h3>Shadow DOMツリー</h3>
    <div id="host"></div>
    <ul id="menu">
      <li>項目1</li>
      <li>項目2</li>
      <li>項目3</li>
    </ul>
    <form>
      <input type="button" value="Shadow DOMにアクセス" id="setCSS">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

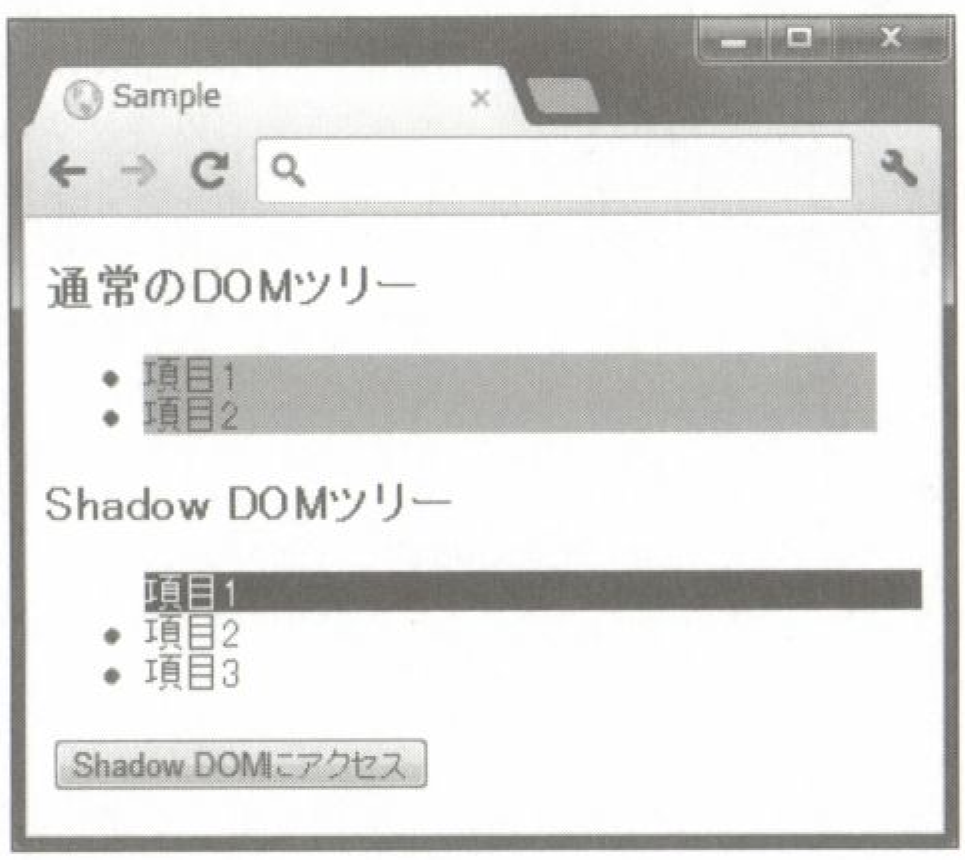
```
window.addEventListener("load", function(){
  // Shadow DOMのホストとなる要素を読み出し
  var shadowHost = document.getElementById("host");
  // Shadow DOMのルートを指定
  var shadowRoot = new WebKitShadowRoot(shadowHost);
  // Shadow DOMに連結するリスト項目
  var myList = document.getElementById("menu");
  // Shadow DOMに追加
  shadowRoot.appendChild(myList);
```

```
  // ボタンにイベントを設定
  document.getElementById("setCSS").addEventListener("click", function(){
    var li = shadowRoot.getElementsByTagName("li");
    li[0].style.color = "white";
    li[0].style.backgroundColor = "black";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、2つ目のリストがShadow DOMとなっています。このため、通常のCSSは反映されていません。



ボタンをクリックすると、Shadow DOM側にある最初のリスト項目の文字色と背景色が変化します。



ONEPOINT **Shadow DOMにある要素にアクセスするには DOM関連のメソッドを使う**

Shadow DOMのルート要素を基準としてDOM関連のメソッドを使うことで、Shadow DOMにある要素にアクセスすることができます。一般的には、「document.getElementsByTagName()」のように記述する部分の「document」をShadow DOMのルートとなる要素(オブジェクト)にします。つまり、基準となるルート要素を変更するだけでよいことになります。

COLUMN Chromeでの設定

Google Chrome 21では標準でShadow DOMが動作しません。Shadow DOMを動作させるには、アドレスバーに「about:flags」と入力し、表示される設定画面で「Shadow DOMを有効にする」の項目で機能するように設定を行います。

関連項目 ▶▶▶

- Shadow DOMとして要素を追加する …… p.388
- 通常のDOMツリーの要素をShadow DOMにする …… p.390

CHAPTER 09

イベント

イベントハンドラを設定する

ここでは、指定した要素にイベントハンドラを設定する方法について解説します。

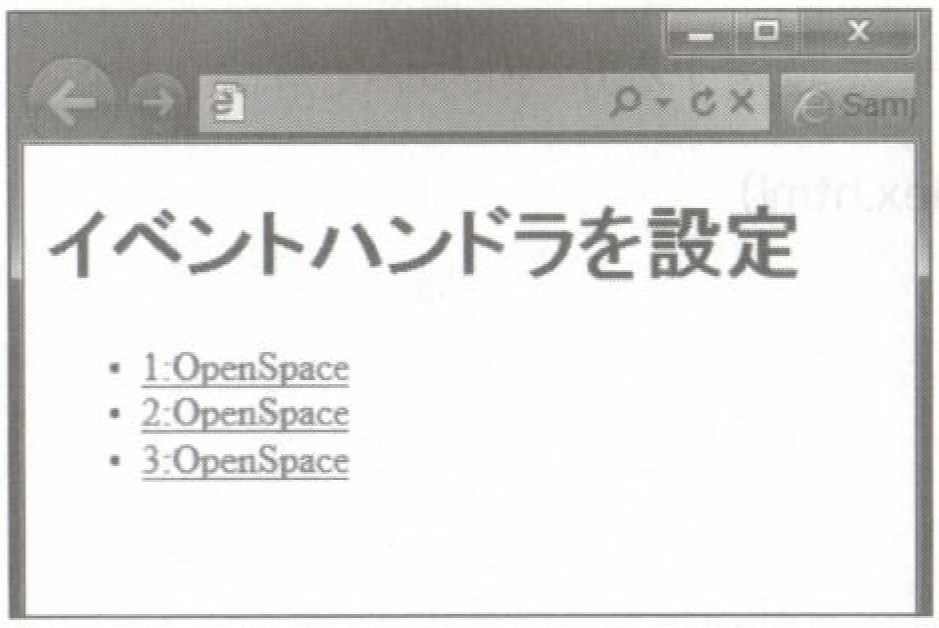
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>イベントハンドラを設定</h1>
    <ul>
      <li><a href="http://www.openspc2.org/">1:OpenSpace</a></li>
      <li><a href="http://www.openspc2.org/">2:OpenSpace</a></li>
      <li><a href="http://www.openspc2.org/">3:OpenSpace</a></li>
    </ul>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.onload = function(){
  // イベントを設定する要素を読み出し
  var ele = document.getElementsByTagName("a");
  // 最初のa要素にイベントハンドラを設定
  ele[0].onclick = function(evt){
    // クリック時の処理
    alert("[1]クリックしました");
    // ブラウザのデフォルトのイベントを禁止
    evt.preventDefault();
    // イベントが伝達されるのを禁止
    evt.stopPropagation();
  }
  // 2番目のa要素にイベントハンドラを設定
  ele[1].onclick = function(evt){
    // クリック時の処理
    alert("[2]クリックしました");
  }
}
```

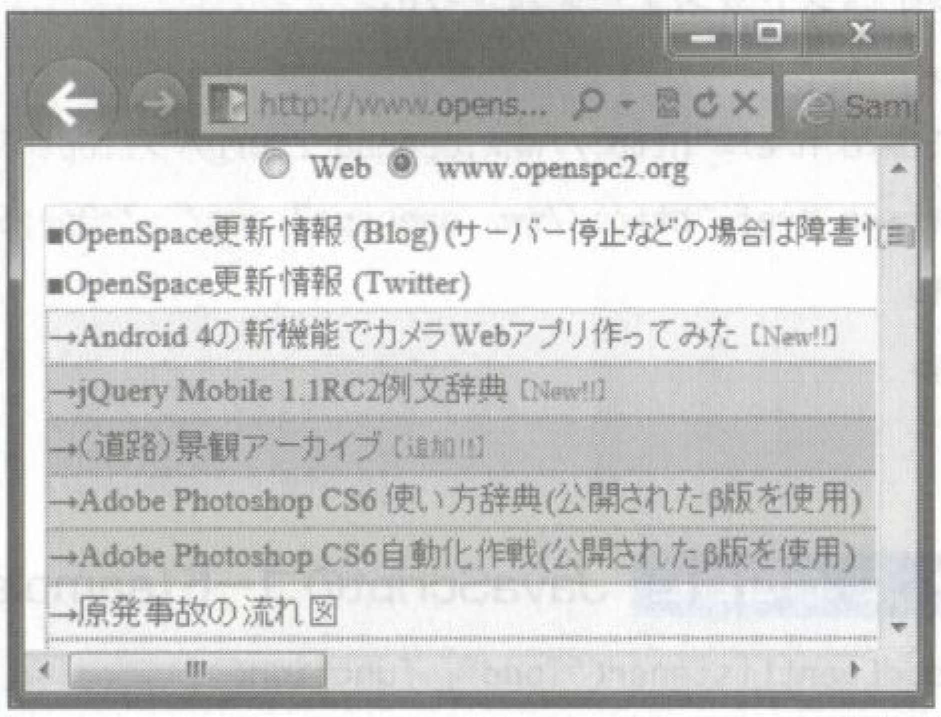
ブラウザでindex.htmlを表示し、最初のリンクをクリックすると設定したイベントハンドラが実行され、アラートダイアログが表示されます。ブラウザのデフォルトのイベントを禁止しているため、リンク先に移動せず、元の状態に戻ります。





2番目のリンクにもイベントハンドラが設定されており、クリックするとアラートダイアログが表示されます。ただし、デフォルトのイベントを禁止していないため、リンク先のページに移動します。





ONEPOINT イベントハンドラを設定するには要素に割り当てられたプロパティを使う

要素には、イベント発生時の処理を設定することができます。処理を設定するには、要素の特定のプロパティにイベントハンドラ(関数)を指定します。設定したいイベント名が「click」なら、「onclick」が要素に割り当てられたプロパティになります。

イベントハンドラ内の処理が終わると、ブラウザのデフォルトのイベントが実行されます。「a」要素であれば指定したリンク先に移動、送信ボタンであればフォームのデータを送信します。このようなブラウザのデフォルトのイベントを実行したくない場合は、イベントオブジェクトの「preventDefault()」メソッドを使います(IEはバージョン9以降)。また、イベントを伝達したくない場合には、イベントオブジェクトの「stopPropagation()」メソッドを使います。

なお、すでにイベントハンドラが設定されている場合、新たに設定すると既存の処理は上書きされてしまいます。上書きせずにイベント処理を追加したい場合は、イベントリスナーを使います(398ページ参照)。

関連項目 ▶▶▶

●イベントリスナーを設定する ……………… p.398

SECTION-173

イベントリスナーを設定する

ここでは、指定した要素にイベントリスナーを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>イベントリスナーを設定</h1>
    <ul>
      <li><a href="http://www.openspc2.org/">1:OpenSpace</a></li>
      <li><a href="http://www.openspc2.org/">2:OpenSpace</a></li>
    </ul>
  </body>
</html>
```

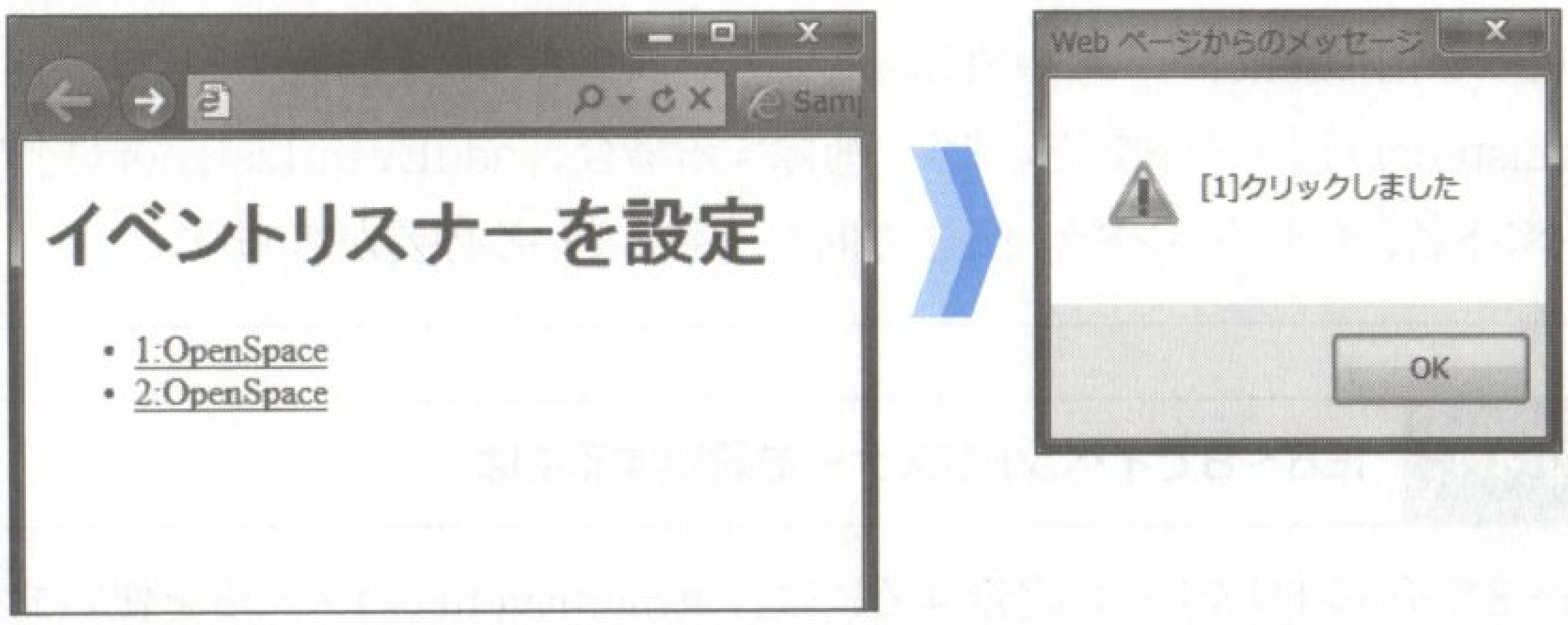
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // イベントを設定する要素を読み出し
  var ele = document.getElementsByTagName("a");
  // 最初のa要素にイベントリスナーを設定
  ele[0].addEventListener("click", function(evt){
    // クリック時の処理
    alert("[1]クリックしました");
    // ブラウザのデフォルトのイベントを禁止
    evt.preventDefault();
    // イベントが伝達されるのを禁止
    evt.stopPropagation();
  }, false);
  // 最初のa要素にさらにイベントリスナーを設定
  ele[0].addEventListener("click", function(evt){
    alert("さらにクリックしました");
  }, false);
}, false);
```

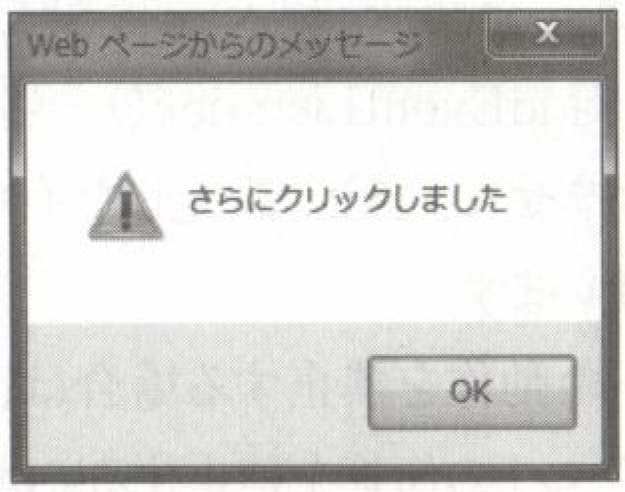
HINT

IE6～8でイベントリスナーを設定する方法は、400ページのCOLUMNを参照してください。

最初のリンクにはイベントリスナーを2回設定されています。ブラウザでindex.htmlを表示し、最初のリンクをクリックすると最初に設定したイベント処理が行われます。ここではダイアログが表示されます。



さらに、設定した2つめのイベント処理が行われ、ダイアログが表示されます。



ONEPOINT イベントリスナーを設定するには「addEventListener()」メソッドを使う

要素には、イベント発生時の処理を設定することができます。要素に用意された「onclick」などのプロパティにイベント発生時の処理を指定することができますが、多人数での開発やライブラリ使用時には既存の処理が上書きされてしまうことがあり、問題が発生する可能性が高くなります。そのような場合はイベントリスナーを使います。イベントリスナーは、既存のイベントハンドラを上書きせずに追加していきます。

イベントリスナーは、要素の「addEventListener()」メソッドを使って行います。最初のパラメータがイベント名になります。これは「click」であれば「"click"」の文字列を指定します。文字列の先頭に「on」の文字を付加する必要はありません。

2番目のパラメータは、イベントハンドラになります。イベントハンドラには、イベントオブジェクトがパラメータとして渡されます。

3番目のパラメータは、イベントの伝達方向になります。「true」、または、「false」を設定します。Google ChromeやSafari、Firefoxでは、3番目のパラメータは省略することができます。

「addEventListener()」メソッドによって追加されたイベントリスナーは、追加した順番に実行されていきます。

イベントが発生しイベントハンドラ内の処理が終わると、ブラウザのデフォルトのイベントが実行されます。「a」要素であれば指定したリンク先に移動、送信ボタンであればフォームのデータを送信します。このようなブラウザのデフォルトのイベントを実行したくない場合

は、イベントオブジェクトの「preventDefault()」メソッドを使います（IEはバージョン9以降）。また、イベントを伝達したくない場合は、イベントオブジェクトの「stopPropagation()」メソッドを使います。

「addEventListener()」メソッドによって追加されたイベントを削除するには、「removeEventListener()」メソッドを使います。削除する場合、「addEventListener()」で設定したイベント名、イベントハンドラ、伝達方向が一致している必要があります。

COLUMN **IE6～8でイベントリスナーを設定するには**

IE6～8でイベントリスナーを設定するには、「attachEvent()」メソッドを使います。最初のパラメータにイベント名を指定しますが、先頭に「on」の文字を付加する必要があります。つまり、「click」イベントなら「onclick」という文字列を指定します。2番目にはイベントハンドラを設定します。なお、「addEventListener()」メソッドと異なり、3番目のパラメータ（イベントの伝達方向）はありません。また、設定したイベントリスナーを削除するには、「detachEvent()」メソッドを使います。

ブラウザのデフォルトイベントの動作を禁止する場合は、「event.returnValue」プロパティに「false」を設定します。イベントの伝達を禁止するには、「event.cancelBubble」プロパティに「true」を設定します。サンプルをIE6～8用に書き換えると、次のようになります。

```
window.attachEvent("onload", function(){
  // イベントを設定する要素を読み出し
  var ele = document.getElementsByTagName("a");
  // 最初のa要素にイベントリスナーを設定
  ele[0].attachEvent("onclick", function(){
    // クリック時の処理
    alert("[1]クリックしました");
    // ブラウザのデフォルトのイベントを禁止
    event.returnValue = false;
    // イベントが伝達されるのを禁止
    event.cancelBubble = true;
  });
  // 最初のa要素にさらにイベントリスナーを設定
  ele[0].attachEvent("onclick", function(evt){
    alert("さらにクリックしました");
  });
});
```

関連項目 ▶▶▶

●イベントハンドラを設定する ………………………………………… p.396

SECTION-174

押されたキーを調べる

ここでは、押されたキーを調べたり、特殊キーの押下状態を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>キー入力</h1>
    <p>何かキーを押して下さい</p>
    <output></output>
  </body>
</html>
```

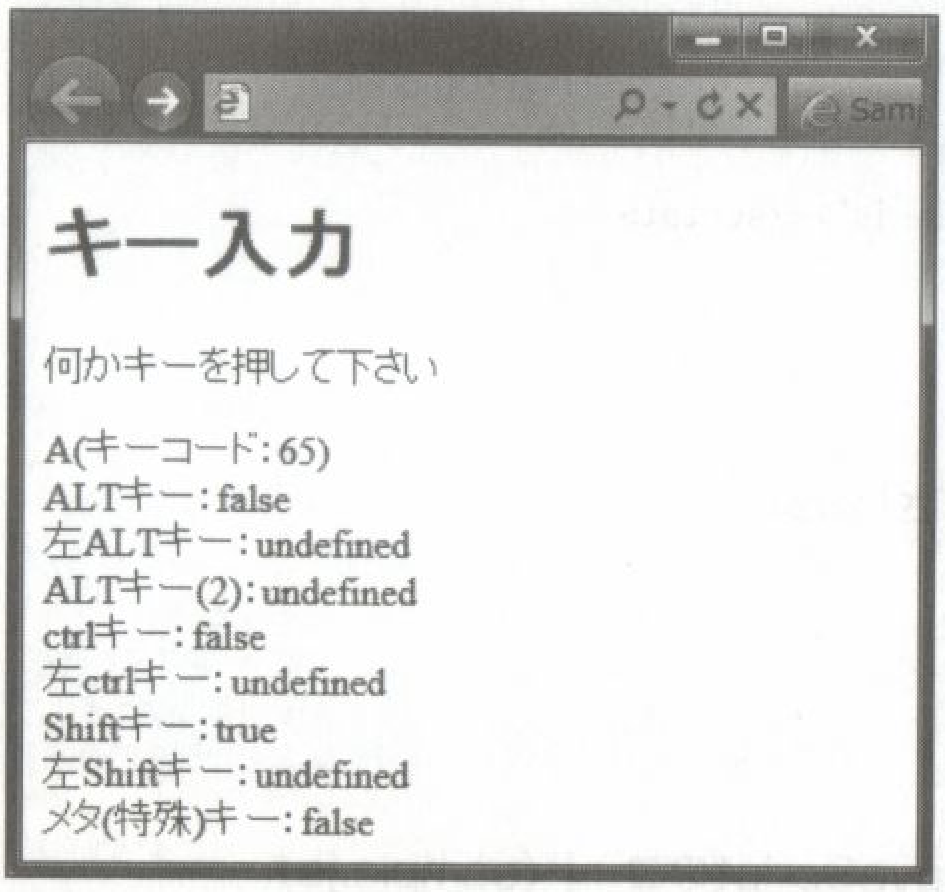
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // body要素にイベントリスナーを設定
  document.body.addEventListener("keydown", function(evt){
    // 押されたキーコードを求める
    var kc = evt.keyCode;
    // キーコードから文字に変換
    var str = String.fromCharCode(kc);
    // 結果を出力
    ele.innerHTML = str+"(キーコード:"+kc+")";
    // 特殊キーの押下状態をチェック
    var altKey = evt.altKey;
    var altKeyLeft = evt.altLeft;
    var altKeyGraph = evt.altGraphKey;
    var ctrlKey = evt.ctrlKey;
    var ctrlKeyLeft = evt.ctrlLeft;
    var shiftKey = evt.shiftKey;
    var shiftKeyLeft = evt.shiftLeft;
    var metaKey = evt.metaKey;
    // 結果を出力
    ele.innerHTML += "<br>ALTキー:"+altKey;
    ele.innerHTML += "<br>左ALTキー:"+altKeyLeft;
    ele.innerHTML += "<br>ALTキー(2):"+altKeyGraph;
    ele.innerHTML += "<br>ctrlキー:"+ctrlKey;
```

▼

```
      ele.innerHTML += "<br>左ctrlキー:"+ctrlKeyLeft;
      ele.innerHTML += "<br>Shiftキー:"+shiftKey;
      ele.innerHTML += "<br>左Shiftキー:"+shiftKeyLeft;
      ele.innerHTML += "<br>メタ(特殊)キー:"+metaKey;
   }, false);
}, false);
```

ブラウザでindex.htmlを表示し、キーボードのキーを押すと、押されたキーのコードや、Shiftキーなど修飾キーなどの押下状態が表示されます。



ONEPOINT

押されたキーを調べるには「keyCode」プロパティや修飾キー専用のプロパティを使う

キー入力に関するイベントは、次の3つがあります。

キーイベント	説明
keydown	キーが押されたとき
keypress	キーが押されたままのとき
keyup	キーが離されたとき

押されたキーを調べるには、イベントオブジェクトの「keyCode」プロパティを読み出します。「keyCode」プロパティが返す値は「keypress」イベントだけ、「keydown」イベントと「keyup」イベントとは異なる値になります。

通常のキーではなく、Shiftキーなどの押下状態を調べる場合には、次の表のような専用のプロパティを読み出します。これらのプロパティはキーが押されると「true」、離されると「false」になります。ただし、ブラウザやバージョンによって利用できるプロパティが異なります。

プロパティ	説明
altKey	左右のAltキー押下状態
altLeft	左のAltキー押下状態(IE6～8)
altGraphKey	左のAltキー押下状態(Google Chrome)
ctrlKey	左右のCtrlキー押下状態
ctrlLeft	左のCtrlキー押下状態(IE6～8)
shiftKey	左右のShiftキー押下状態

プロパティ	説明
shiftLeft	左のShiftキー押下状態(IE6～8)
metaKey	メタキー(Macの場合はコマンドキー)

なお、IE、Google Chrome、Safariでは、直接、「event」オブジェクトを参照することもできます。

COLUMN スマートフォンでのキー入力

スマートフォンでは、ハード的にキーボードがない端末がほとんどです。このため、パソコンのキーボードのようにキー入力処理を行うことはできません。キー入力を行うためには、テキストフィールドなどを用意し、ソフトウェアキーボードを利用して入力してもらうことになります。

iOS 5では英語キーボード表示の場合、リアルタイムにキーコードを取得することができます。ただし、日本語キーボード表示の場合、キーコードは「0」になります。

Androidでもソフトウェアキーボードを使って入力されたキーの情報を取得することができます。Androidも英語キーボード表示の場合、リアルタイムにキーコードを取得できますが、日本語キーボード表示の場合は何も返しません(キーコードが入らない)。

Windows Phone(7.5)も英語キーボード表示の場合、リアルタイムにキーコードを取得できますが、日本語キーボード表示の場合、押したキーと「keyCode」プロパティで返されるキー(値)は一致しません(「あ」がa、「か」がb、「さ」がc、「た」がdになる)。

SECTION-175

イベントが発生した座標を求める

ここでは、イベントが発生した座標を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

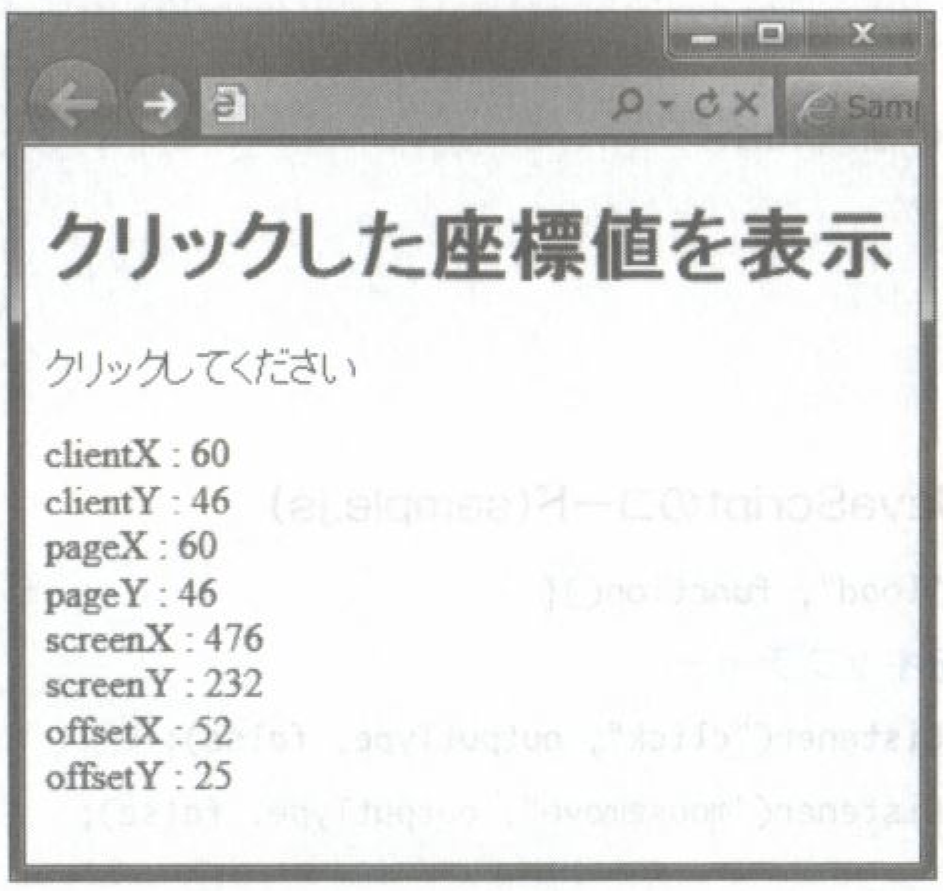
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>クリックした座標値を表示</h1>
    <p>クリックしてください</p>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // body要素にイベントリスナーを設定
  window.addEventListener("click", coords, false);
  // h1要素にイベントリスナーを設定
  var h1 = document.getElementsByTagName("output")[0];
  h1.addEventListener("click", coords, false);
}, false);
// 座標表示
function coords(evt){
  var ele = document.getElementsByTagName("output")[0];
  // クライアント座標を求める
  var cx = evt.clientX;
  var cy = evt.clientY;
  // ページ座標を求める
  var px = evt.pageX;
  var py = evt.pageY;
  // モニタ画面上の座標を求める
  var sx = evt.screenX;
  var sy = evt.screenY;
  // オフセット座標を求める
  var ox = evt.offsetX;
  var oy = evt.offsetY;
  // 結果を出力
  ele.innerHTML = "clientX : "+cx+"<br>";
```

```
    ele.innerHTML += "clientY : "+cy+"<br>";
    ele.innerHTML += "pageX : "+px+"<br>";
    ele.innerHTML += "pageY : "+py+"<br>";
    ele.innerHTML += "screenX : "+sx+"<br>";
    ele.innerHTML += "screenY : "+sy+"<br>";
    ele.innerHTML += "offsetX : "+ox+"<br>";
    ele.innerHTML += "offsetY : "+oy+"<br>";
    // イベントの伝達を禁止
    evt.stopPropagation();
}
```

ブラウザでindex.htmlを表示し、ブラウザのウィンドウ内をクリックすると、クリックした場所の座標値が表示されます。



ONEPOINT イベントが発生した座標を求めるには「clientX」プロパティや「clientY」プロパティなどを使う

イベントが発生すると、イベントオブジェクトの各プロパティに座標値が格納されます。求める座標の種類に応じてプロパティが用意されています。座標値はすべてピクセル値で返されます。

プロパティ	説明
clientX	要素のX座標
clientY	要素のY座標
pageX	ページ上のX座標(スクロール量が加算される)
pageY	ページ上のY座標(スクロール量が加算される)
screenX	モニタ画面上でのX座標
screenY	モニタ画面上でのY座標
offsetX	要素からのX座標オフセット
offsetY	要素からのY座標オフセット

SECTION-176

発生したイベントの種類を求める

ここでは、発生したイベントの種類を求める方法について解説します。

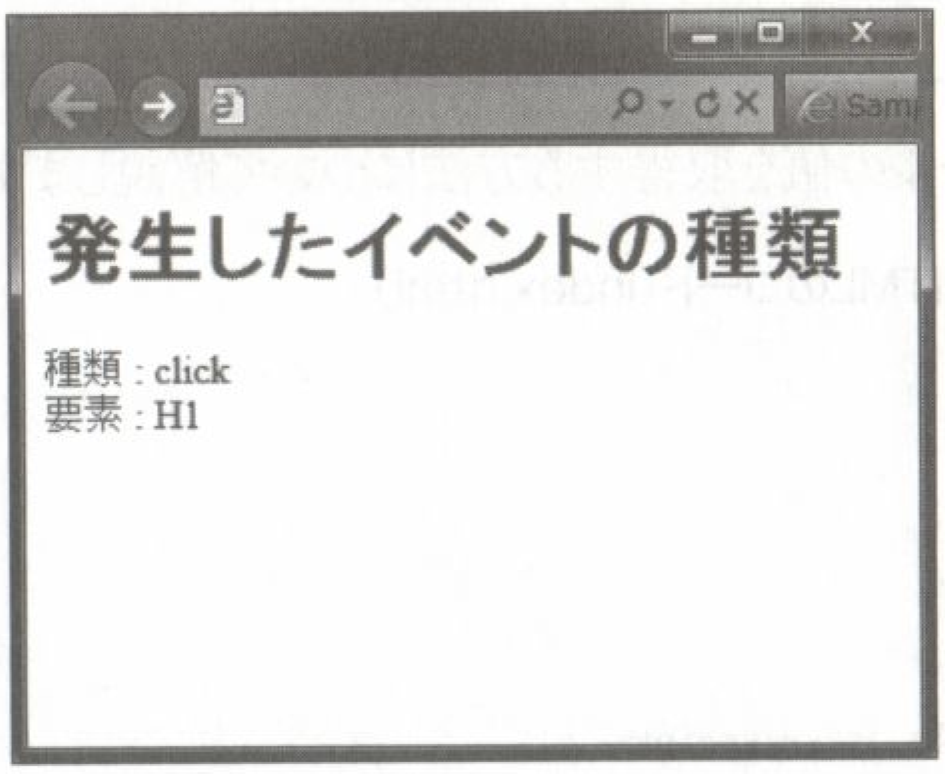
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>発生したイベントの種類</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // body要素のイベントをキャプチャー
  document.body.addEventListener("click", outputType, false);
  document.body.addEventListener("mousemove", outputType, false);
}, false);
// イベントの種類を表示
function outputType(evt){
  var ele = document.getElementsByTagName("output")[0];
  // イベントの種類を求める
  var evtType = evt.type;
  // イベントが発生した要素
  var evtElelement = evt.target.tagName;
  // 結果を出力
  ele.innerHTML = "種類 : "+evtType+"<br>";
  ele.innerHTML += "要素 : "+evtElelement+"<br>";
  // イベントの伝達を禁止
  evt.stopPropagation();
}
```

ブラウザでindex.htmlを表示し、ページ上にマウスを移動させるかクリックすると、イベントの種類と発生したタグ名が表示されます。



なお、ブラウザによっては、画面書き換えのタイミングや処理速度が速すぎるために、種類のところに「click」が表示されない場合があります。その場合は、「console.log()」メソッドを使って変数「evtType」を出力すると、「click」イベントが発生していることを確認できます(346ページ参照)。

ONEPOINT **発生したイベントの種類を求めるには「type」プロパティを使う**

発生したイベントの種類を求めるには、イベントオブジェクトの「type」プロパティを参照します。また、「target」プロパティにはイベントが発生した要素への参照が入っているため、要素名や要素そのものを操作することもできます。

なお、発生したイベントに応じて処理を振り分けることもできます。主なイベントは、次の表のようになります。

イベント	説明
click	クリック
dblclick	ダブルクリック
mouseover	マウスオーバー
mouseout	マウスアウト
mousemove	マウス移動
keydown	キーが押された
keyup	キーが離された

SECTION-177

マウスホイールの値を取得する

ここでは、マウスホイールの値を取得する方法について解説します。

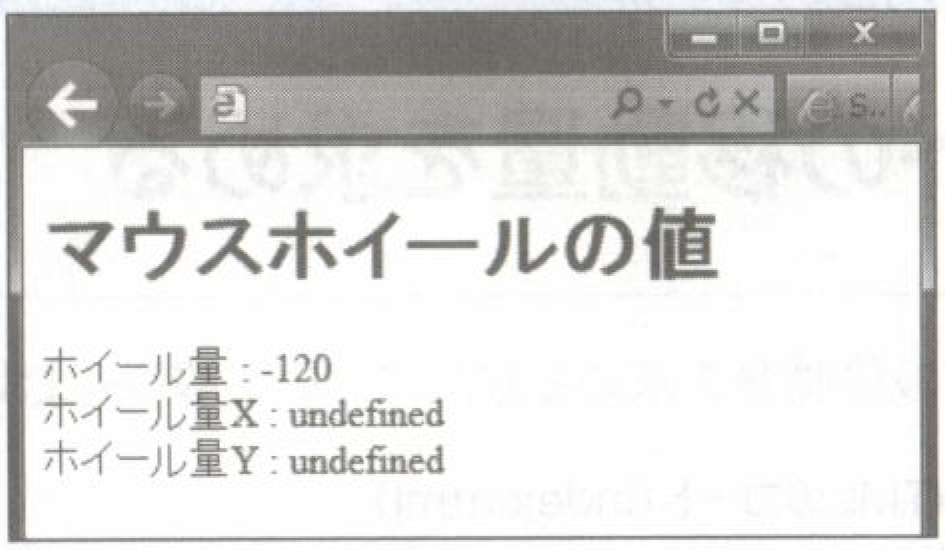
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>マウスホイールの値</h1>
    <output></output>
  </body>
</html>
```

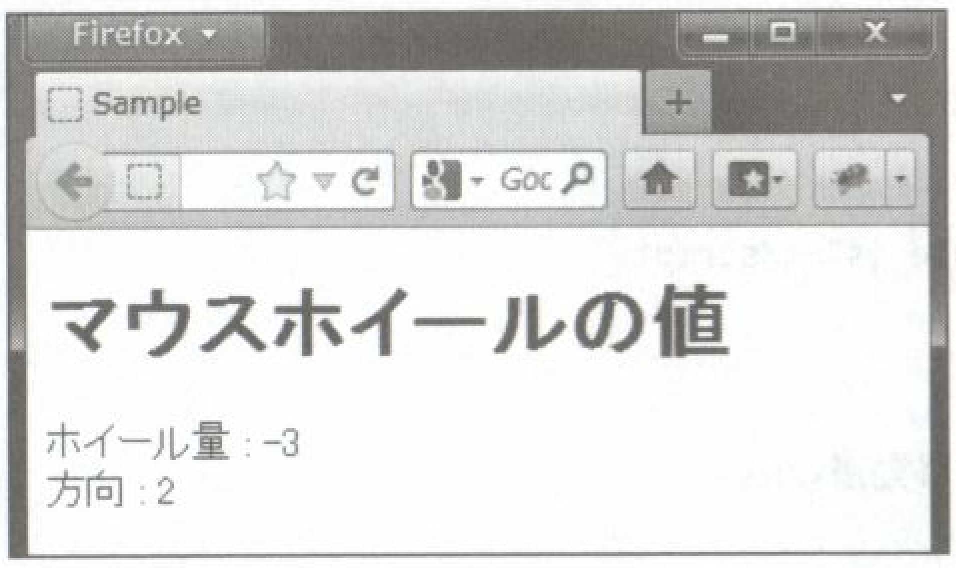
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // body要素のイベントをキャプチャー
  document.body.addEventListener("mousewheel", function(evt){
    // 回転量を求める
    var delta = evt.wheelDelta;
    var deltaX = evt.wheelDeltaX;
    var deltaY = evt.wheelDeltaY;
    // 結果を出力
    ele.innerHTML = "ホイール量 : "+delta+"<br>";
    ele.innerHTML += "ホイール量X : "+deltaX+"<br>";
    ele.innerHTML += "ホイール量Y : "+deltaY;
  }, false);
  document.body.addEventListener("DOMMouseScroll", function(evt){
    // 回転量を求める
    var detail = evt.detail;
    var axis = evt.axis;
    // 結果を出力
    ele.innerHTML = "ホイール量 : "+detail+"<br>";
    ele.innerHTML += "方向 : "+axis;
  }, false);
}, false);
```

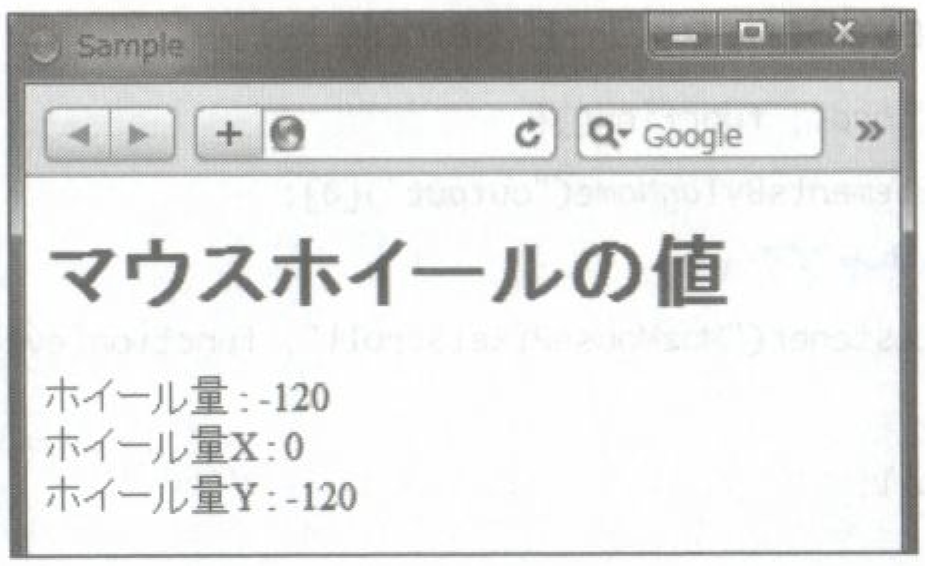
ブラウザでindex.htmlを表示すると、ページ上にマウスホイールの回転量が表示されます。IE9では、回転量のみがわかります。



Firefoxの場合は、回転量と、縦方向、横方向どちらに回転したのかもわかります。



Google ChromeやSafariでは、回転量と、左右の回転量がそれぞれ別のプロパティに入っています。



ONEPOINT **マウスホイールの値を取得するには「mousewheel」イベントや「DOMMouseScroll」イベントを検知する**

Firefox以外で、マウスホイールの値を取得するには、「mousewheel」イベントを検知します。ホイールがどの程度、回転したのかは、イベントハンドラに渡されるイベントオブジェクトの「delta」プロパティに入ります。

Firefoxでは、マウスホイールが回転すると、「DOMMouseScroll」イベントが発生します。ホイールの回転量は、イベントオブジェクトの「detail」プロパティに入ります。

IEでは回転量しかわかりませんが、Firefoxでは縦方向、横方向どちらに回転したかを示す「axis」プロパティがあります。また、Google ChromeやSafariなどのWebKit系では、縦方向と横方向の回転量は、別々のプロパティに入ります。

なお、ブラウザによってはスクロール可能な状態でないと値が取得できない場合があります。

関連項目 ▶▶▶

●トラックパッドの移動量を求める……p.410

SECTION-178

トラックパッドの移動量を求める

ここでは、トラックパッドの移動量を求める方法について解説します。

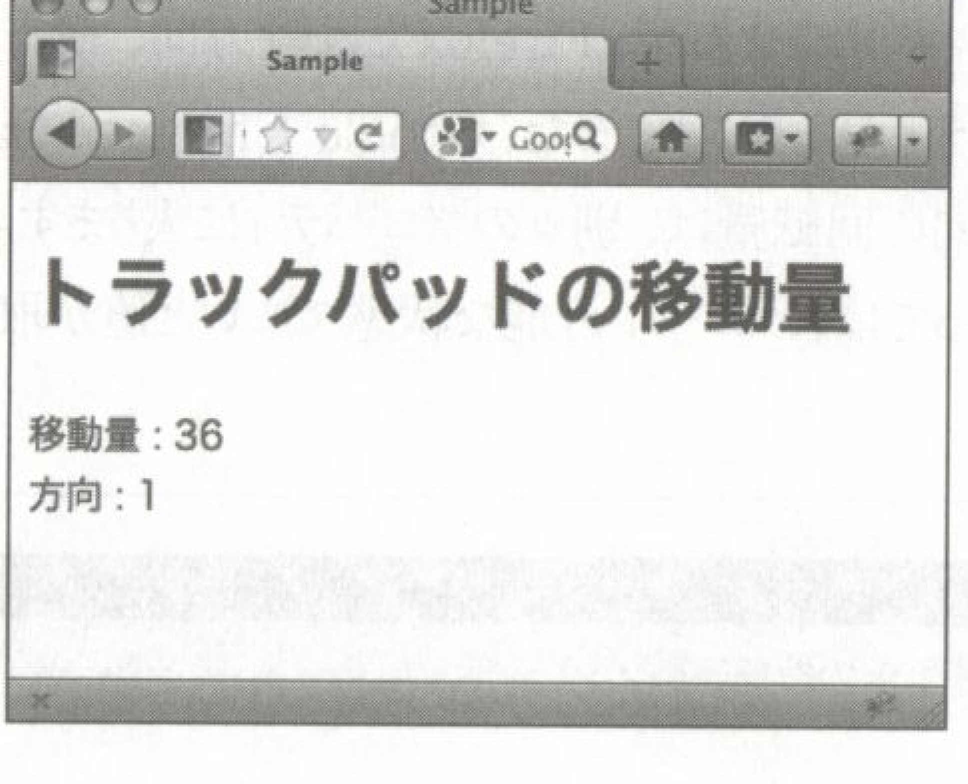
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>トラックパッドの移動量</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // body要素のイベントをキャプチャー
  document.body.addEventListener("MozMousePixelScroll", function(evt){
    // 移動量を求める
    var detail = evt.detail;
    var axis = evt.axis;
    // 結果を出力
    ele.innerHTML = "移動量 : "+detail+"<br>";
    ele.innerHTML += "方向 : "+axis;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、トラックパッドを使って2本指でスクロールさせた場合に、移動量と方向が表示されます。



ONEPOINT トラックパッドの移動量を求めるには「MozMousePixelScroll」イベントを検知する

トラックパッドの移動量を求めるには、「MozMousePixelScroll」イベントを検知します。トラックパッドでどの程度、移動させたのかは、イベントハンドラに渡されるイベントオブジェクトの「delta」プロパティに入ります。

トラックパッドで縦方向に2本指を動かしたのか、横方向に動かしたのかは、イベントオブジェクトの「axis」プロパティに入ります。「axis」プロパティの値は、次のようになります。

値	説明
1	横方向の場合
2	縦方向の場合

COLUMN マウスホイールとの関係

「MozMousePixelScroll」イベントは、トラックパッドだけでなく、マウスホイールの値が変化した場合でも発生します。この場合でも、「detail」プロパティに移動量、「axis」プロパティに移動方向が入ります。

関連項目 ▶▶▶

● マウスホイールの値を取得する ………………………………………………………… p.408

SECTION-179

フォーカスした際に処理を行う

ここでは、フォーカスした際に処理を行う方法とフォーカスが外れた場合に処理を行う方法について解説します。

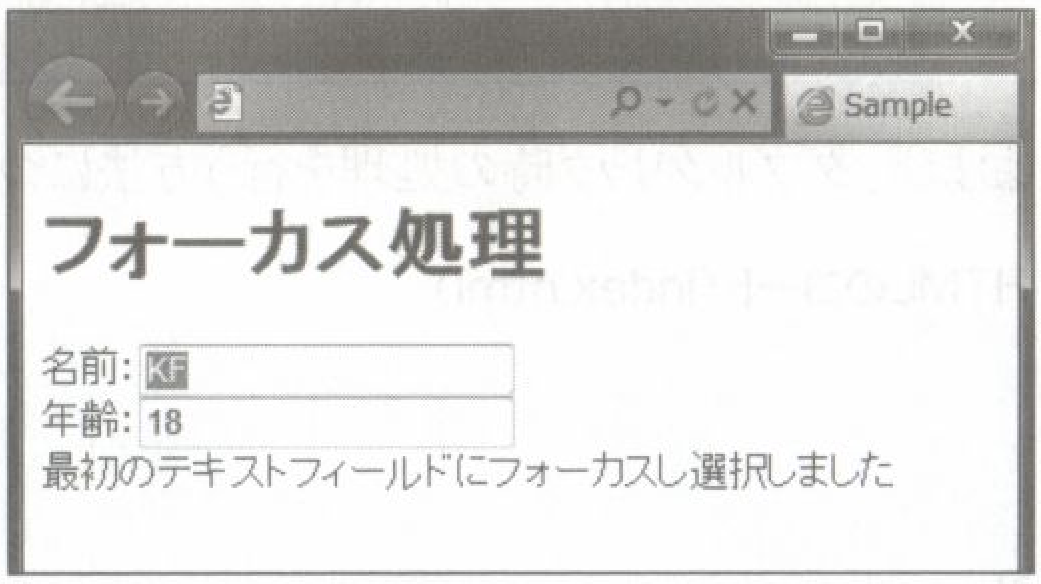
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>フォーカス処理</h1>
    <form>
      名前：<input type="text" value="KF" id="uName"><br>
      年齢：<input type="text" value="18" id="uAge">
    </form>
    <output></output>
  </body>
</html>
```

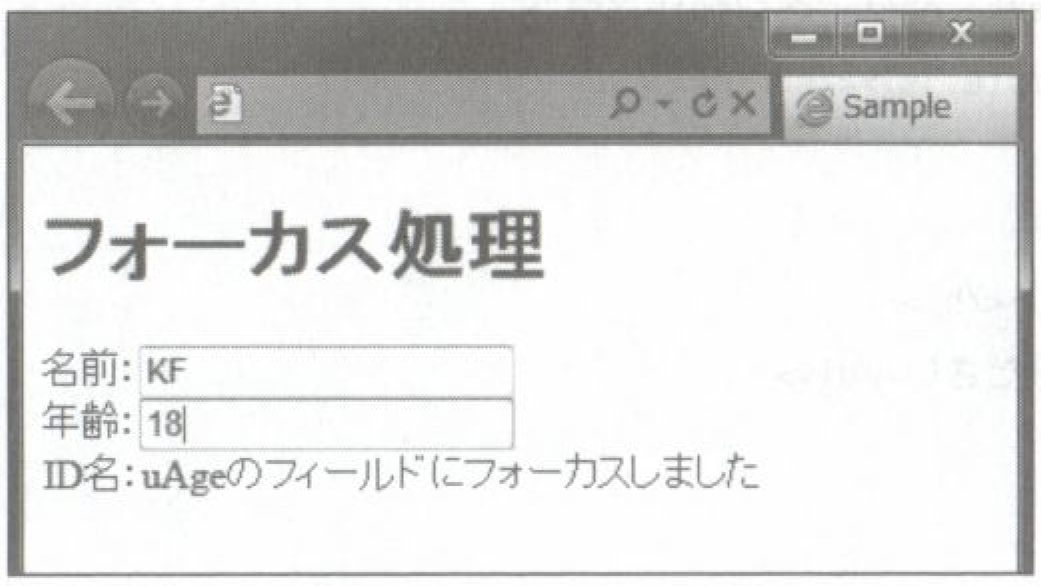
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 最初のテキストフィールドにフォーカス
  var fld1 = document.getElementById("uName");
  fld1.focus();
  // さらに選択。その旨メッセージを表示
  fld1.select();
  ele.innerHTML = "最初のテキストフィールドにフォーカスし選択しました";
  // 2番目のテキストフィールドにフォーカスした場合の処理
  var fld2 = document.getElementById("uAge");
  fld2.addEventListener("focus", function(evt){
    var targetID = evt.target.id;
    ele.innerHTML = "ID名："+targetID+"のフィールドにフォーカスしました";
  }, false);
  // 2番目のテキストフィールドのフォーカスが外れた場合の処理
  fld2.addEventListener("blur", function(evt){
    ele.innerHTML = "フォーカスが外れました";
  }, false);
}, false);
```

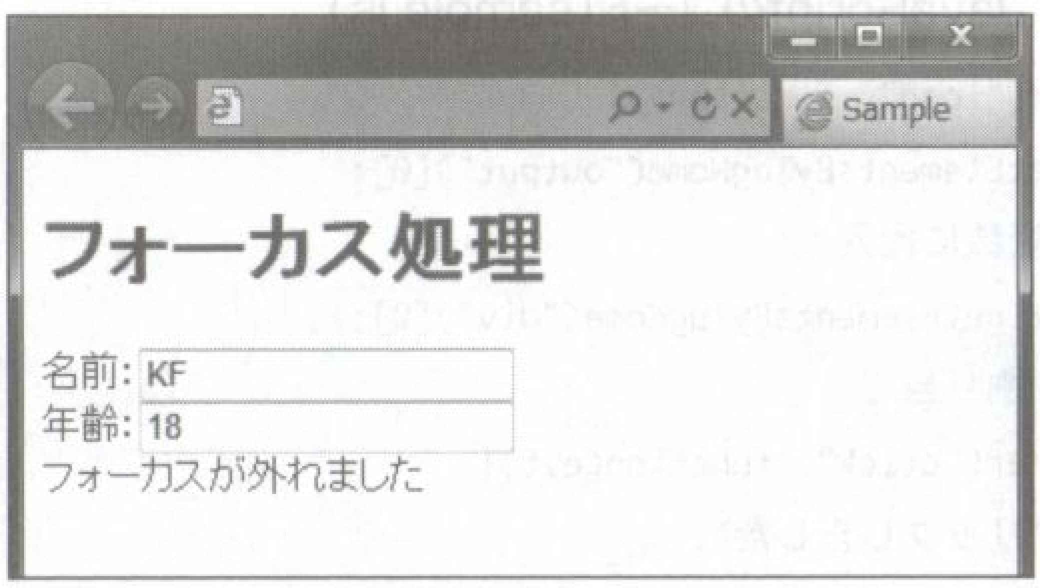
ブラウザでindex.htmlを表示し、ページが読み込まれると、最初のフィールドにフォーカスが移って内容が選択されます。



2番目のフィールドにフォーカスを移すと、フォーカスがされたことを示す文字が表示されます。



フォーカスが外れると、フォーカスが外れたことを示す文字が表示されます。



ONEPOINT フォーカスに関する処理を行うには「focus」イベントや「blur」イベントを使う

ブラウザにはフォーカスを扱うメソッドがいくつか用意されています。特定のテキストフィールドにフォーカスを移す場合は、「focus()」メソッドを使います。テキストフィールドの場合、入力済みの文字を選択状態にしたいのであれば、「focus()」メソッドに続けて「select()」メソッドを呼び出します。

テキストフィールドなどにフォーカスされると、「focus」イベントが発生します。逆にフォーカスが外れると、「blur」イベントが発生します。イベント発生時に処理を行う場合は、イベントリスナーなどを使ってイベントハンドラを設定します。イベントハンドラにはイベントオブジェクトが渡されるので、イベントが発生した要素に対して何か処理を行うこともできます。

関連項目 ▶▶▶

● フォーム要素のフォーカスを設定する ……………………………… p.323

SECTION-180

クリック時/ダブルクリック時の処理を行う

ここでは、クリック時、および、ダブルクリック時の処理を行う方法について解説します。

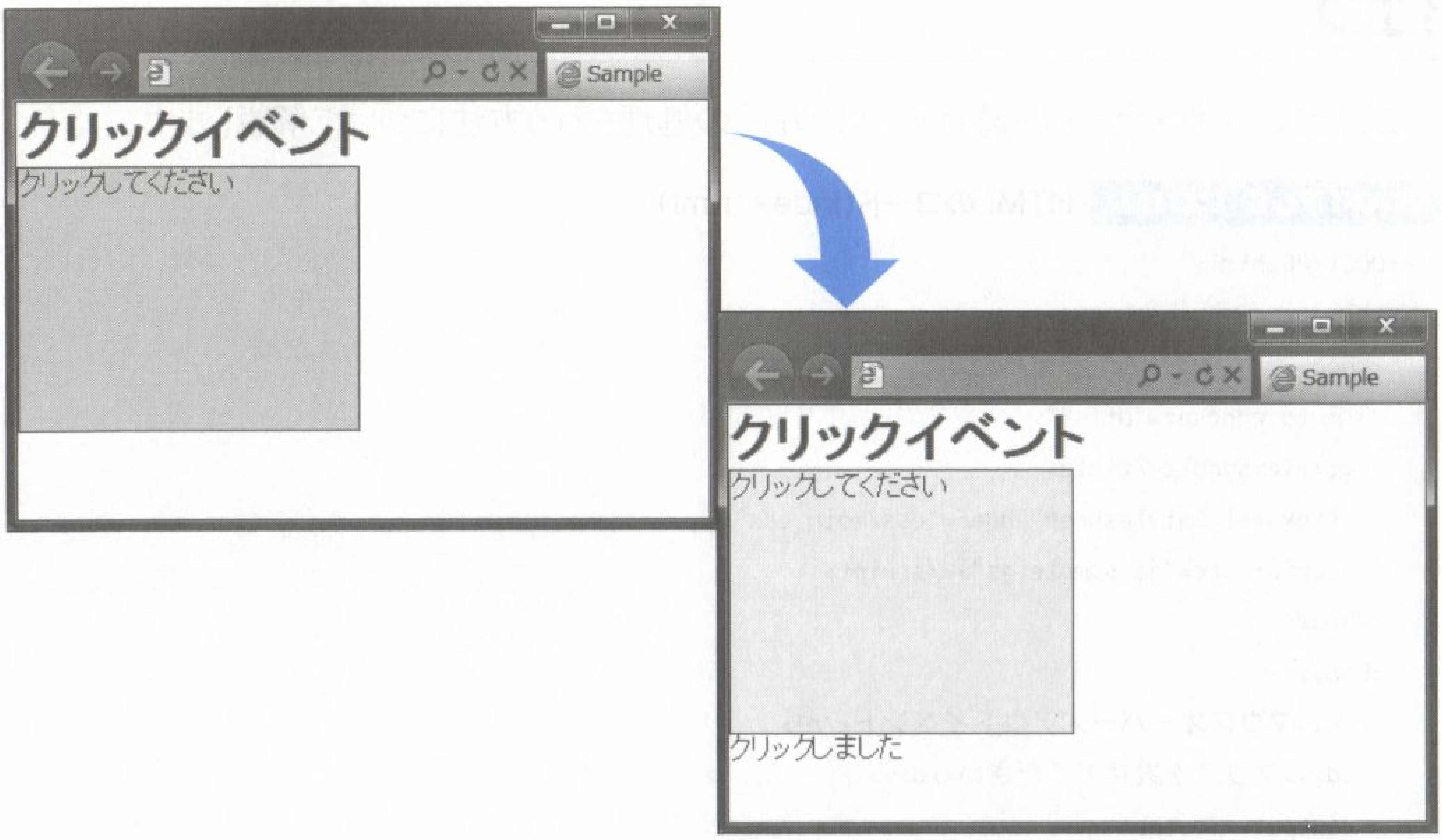
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>クリックイベント</h1>
    <div>クリックしてください</div>
    <output></output>
  </body>
</html>
```

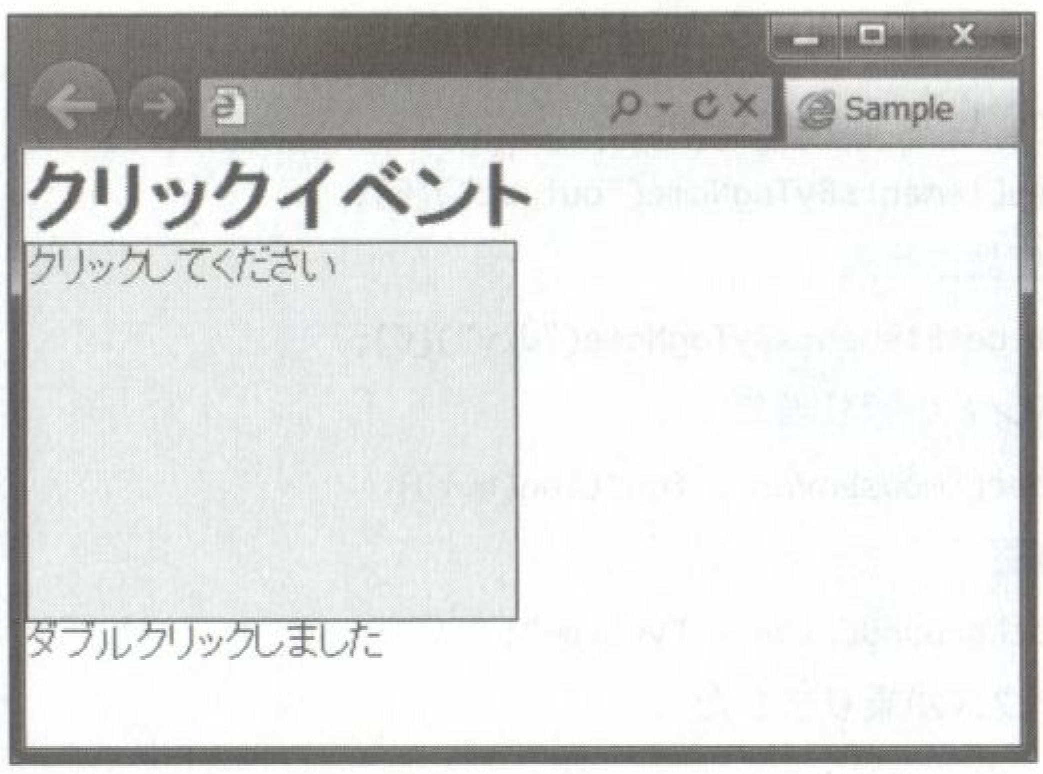
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // div要素への参照を変数に代入
  var myArea = document.getElementsByTagName("div")[0];
  // クリックイベントを割り当て
  myArea.addEventListener("click", function(evt){
    ele.innerHTML = "クリックしました";
    evt.target.style.backgroundColor = "yellow";
  }, false);
  // ダブルクリックイベントを割り当て
  myArea.addEventListener("dblclick", function(evt){
    ele.innerHTML = "ダブルクリックしました";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、画面内の四角い領域をクリックすると、クリックしたことを示すメッセージが表示され、背景色が黄色に変わります。



画面内の四角い領域をダブルクリックすると、ダブルクリックしたことを示すメッセージが表示されます。



ONEPOINT

クリック時に処理を行うには「click」イベント、ダブルクリック時に処理を行うには「dblclick」イベントを捕捉する

クリック時に処理を行うには、「click」イベントを捕捉します。イベントを割り当てる要素に対して、「onclick」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。呼び出されるイベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトには、イベントが発生した要素の情報も入っているため、サンプルのように要素に対して処理を行うこともできます。

クリックでなく、ダブルクリックの場合は、「dblclick」イベントを捕捉します。

SECTION-181

マウスオーバー時/マウスアウト時の処理を行う

ここでは、マウスオーバー時/マウスアウト時の処理を行う方法について解説します。

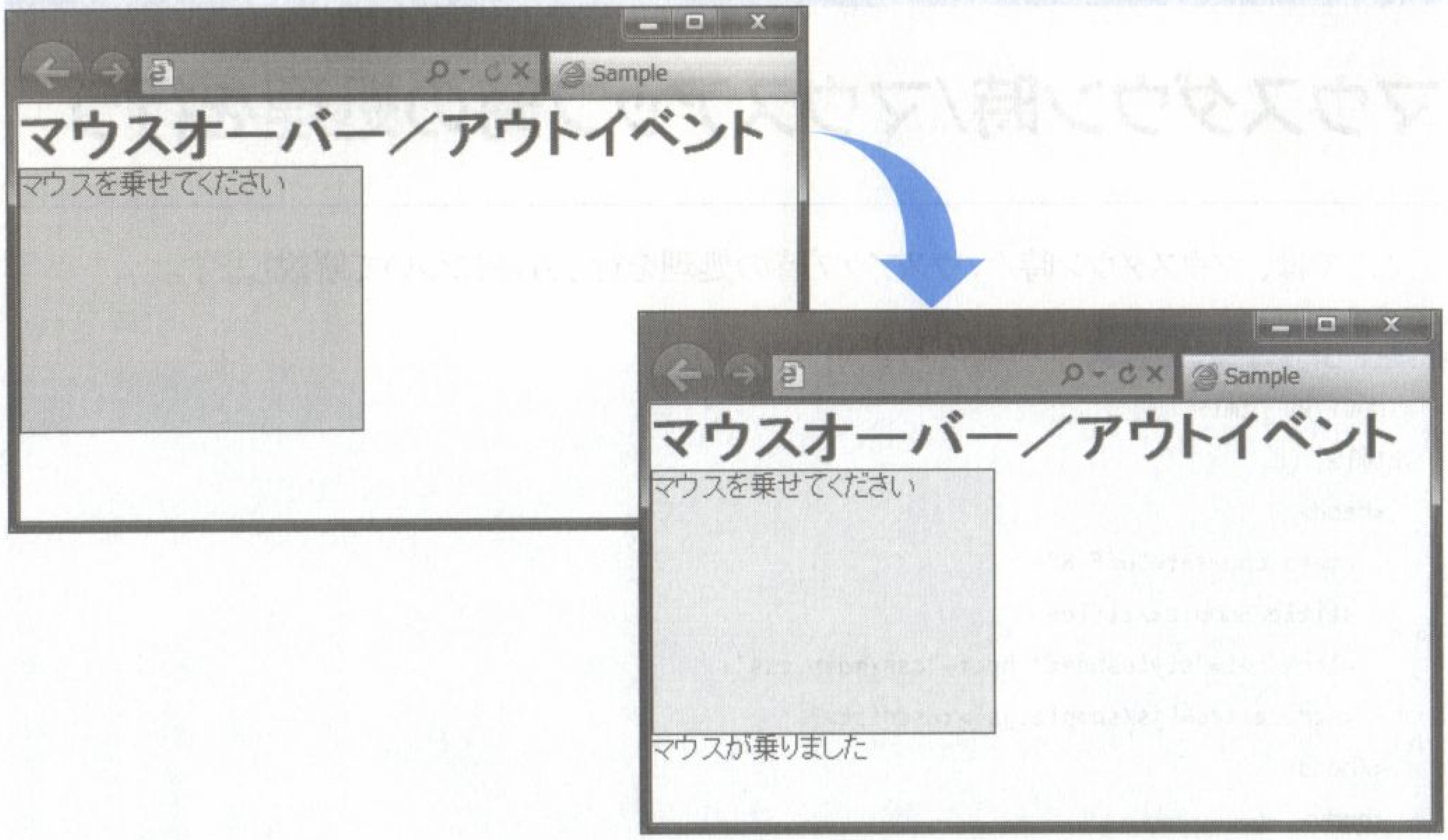
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>マウスオーバー／アウトイベント</h1>
    <div>マウスを乗せてください</div>
    <output></output>
  </body>
</html>
```

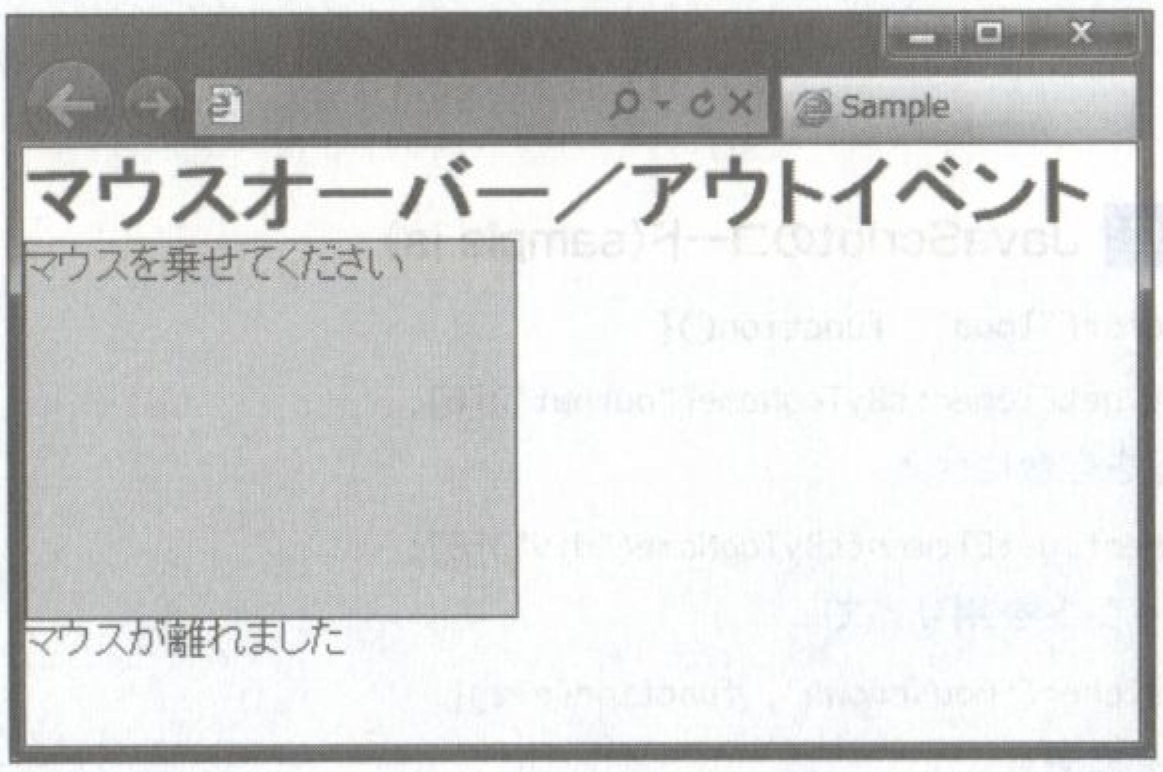
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // div要素への参照を変数に代入
  var myArea = document.getElementsByTagName("div")[0];
  // マウスオーバーイベントを割り当て
  myArea.addEventListener("mouseover", function(evt){
    // 黄色に背景色を変更
    evt.target.style.backgroundColor = "yellow";
    ele.innerHTML = "マウスが乗りました";
  }, false);
  // マウスアウトイベントを割り当て
  myArea.addEventListener("mouseout", function(evt){
    // 黄色を元の色に変更
    evt.target.style.backgroundColor = "#ffd592";
    ele.innerHTML = "マウスが離れました";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、画面内の四角い領域にマウスを重ねると、マウスが乗ったことを示すメッセージが表示され、背景色が黄色に変わります。



マウスが離れると、マウスが離れたことを示すメッセージが表示され、背景色が元の色に戻ります。



なお、Windows Phoneでは、タッチするたびにマウスオーバーとマウスアウトが切り替わります。

ONEPOINT　マウスオーバー時の処理を行うには「mouseover」イベント、マウスアウト時の処理を行うには「mouseout」イベントを捕捉する

マウスオーバー時に処理を行うには、「mouseover」イベントを捕捉します。イベントを割り当てる要素に対して、「onmouseover」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。呼び出されるイベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトにはイベントが発生した要素の情報も入っているため、サンプルのように要素に対して処理を行うこともできます。

マウスアウト時の場合は、「mouseout」イベントを捕捉します。「mouseout」イベントの場合は、要素の「onmouseout」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。

SECTION-182

マウスダウン時/マウスアップ時の処理を行う

ここでは、マウスダウン時/マウスアップ時の処理を行う方法について解説します。

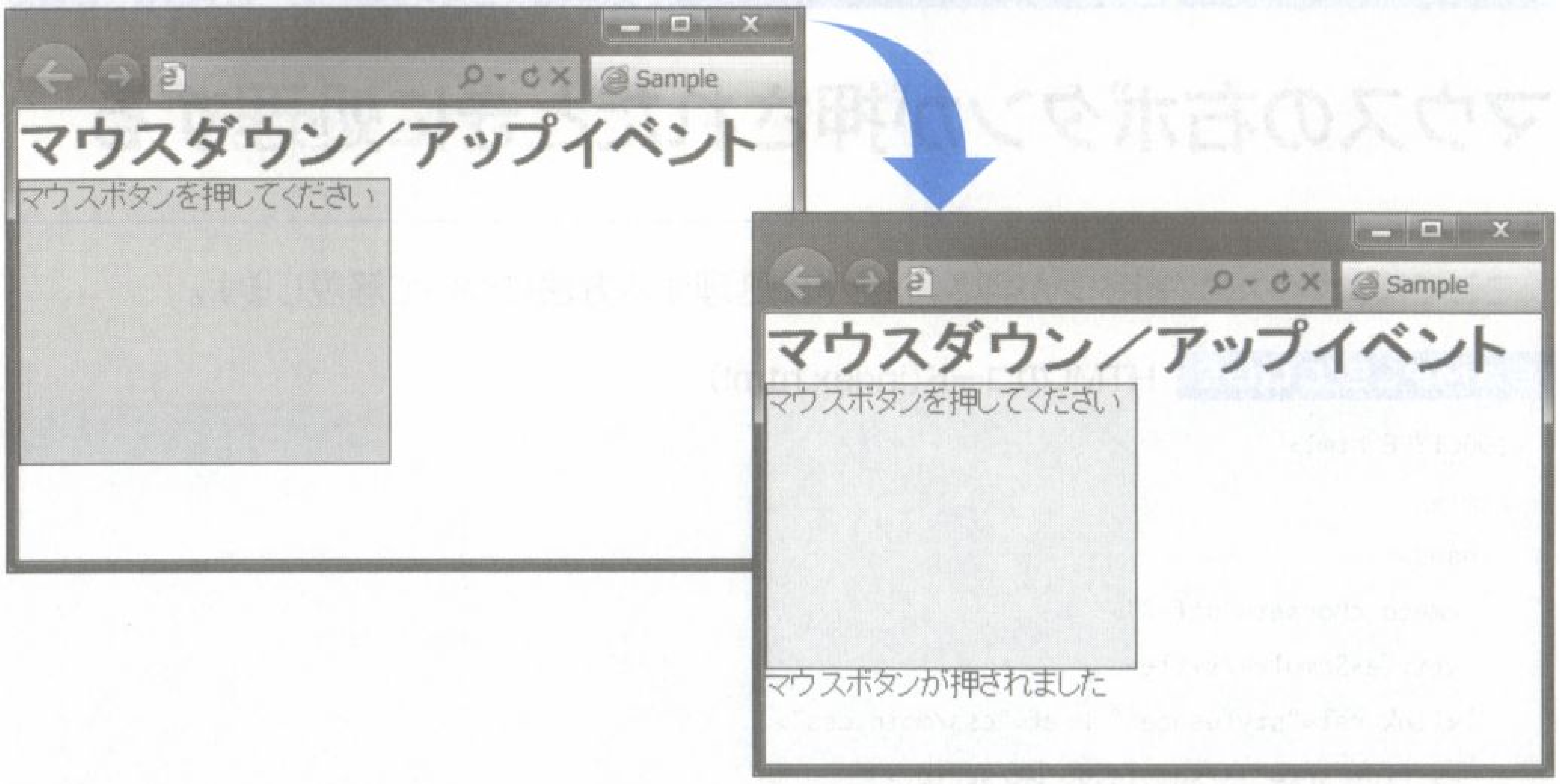
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>マウスダウン／アップイベント</h1>
    <div>マウスボタンを押してください</div>
    <output></output>
  </body>
</html>
```

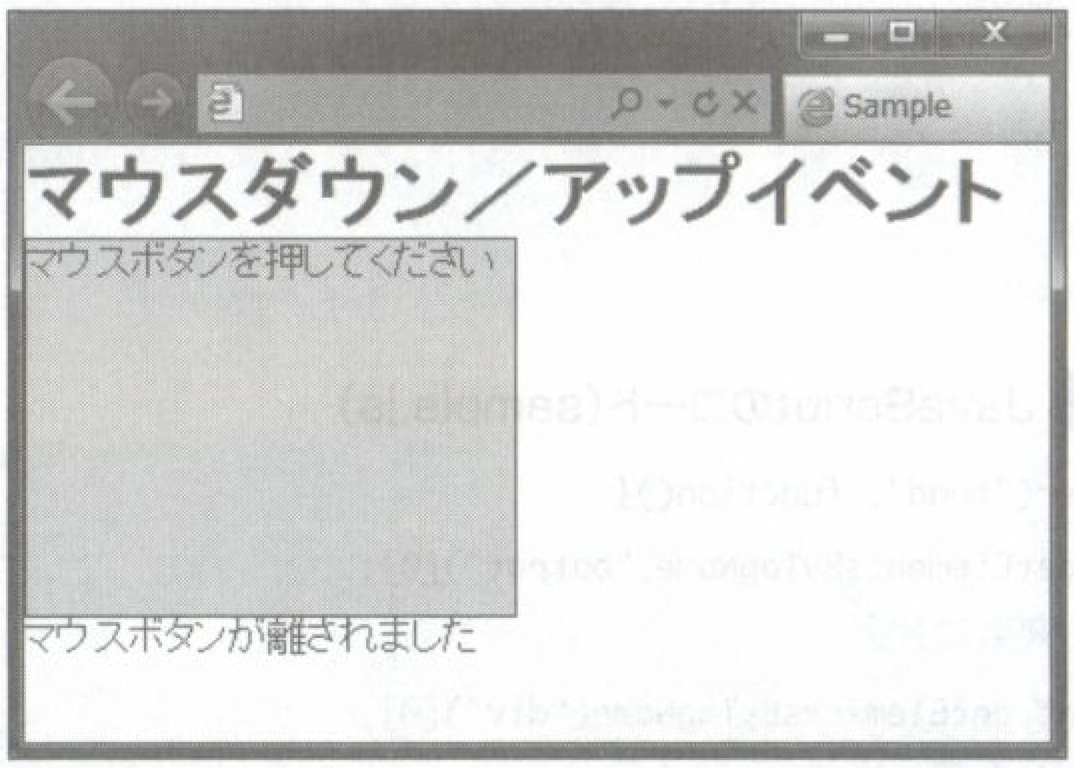
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // div要素への参照を変数に代入
  var myArea = document.getElementsByTagName("div")[0];
  // マウスダウンイベントを割り当て
  myArea.addEventListener("mousedown", function(evt){
    // 黄色に背景色を変更
    evt.target.style.backgroundColor = "yellow";
    ele.innerHTML = "マウスボタンが押されました";
  }, false);
  // マウスアップイベントを割り当て
  myArea.addEventListener("mouseup", function(evt){
    // 黄色を元の色に変更
    evt.target.style.backgroundColor = "#ffd592";
    ele.innerHTML = "マウスボタンが離されました";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、画面内の四角い領域でマウスのボタンを押すと、マウスボタンが押されたことを示すメッセージが表示され、背景色が黄色に変わります。



マウスボタンが離されると、マウスボタンが離されたことを示すメッセージが表示され、背景色が元の色に戻ります。



なお、Windows Phoneでは、マウスアップ時のイベントのみ取得できます。

ONEPOINT　マウスダウン時の処理を行うには「mousedown」イベント、マウスアップ時の処理を行うには「mouseup」イベントを捕捉する

マウスのボタンが押された（マウスダウン）ときの処理を行うには、「mousedown」イベントを捕捉します。イベントを割り当てる要素に対して、「onmousedown」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。呼び出されるイベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトにはイベントが発生した要素の情報も入っているため、サンプルのように要素に対して処理を行うこともできます。

マウスのボタンが離された（マウスアップ）ときの処理を行うには、「mouseup」イベントを捕捉します。「mouseup」イベントの場合は、要素の「onmouseup」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。

関連項目 ▶▶▶

● マウスの右ボタンが押されたときに処理する ……………………………… p.420

SECTION-183

マウスの右ボタンが押されたときに処理する

ここでは、マウスの右ボタンが押されたときに処理する方法について解説します。

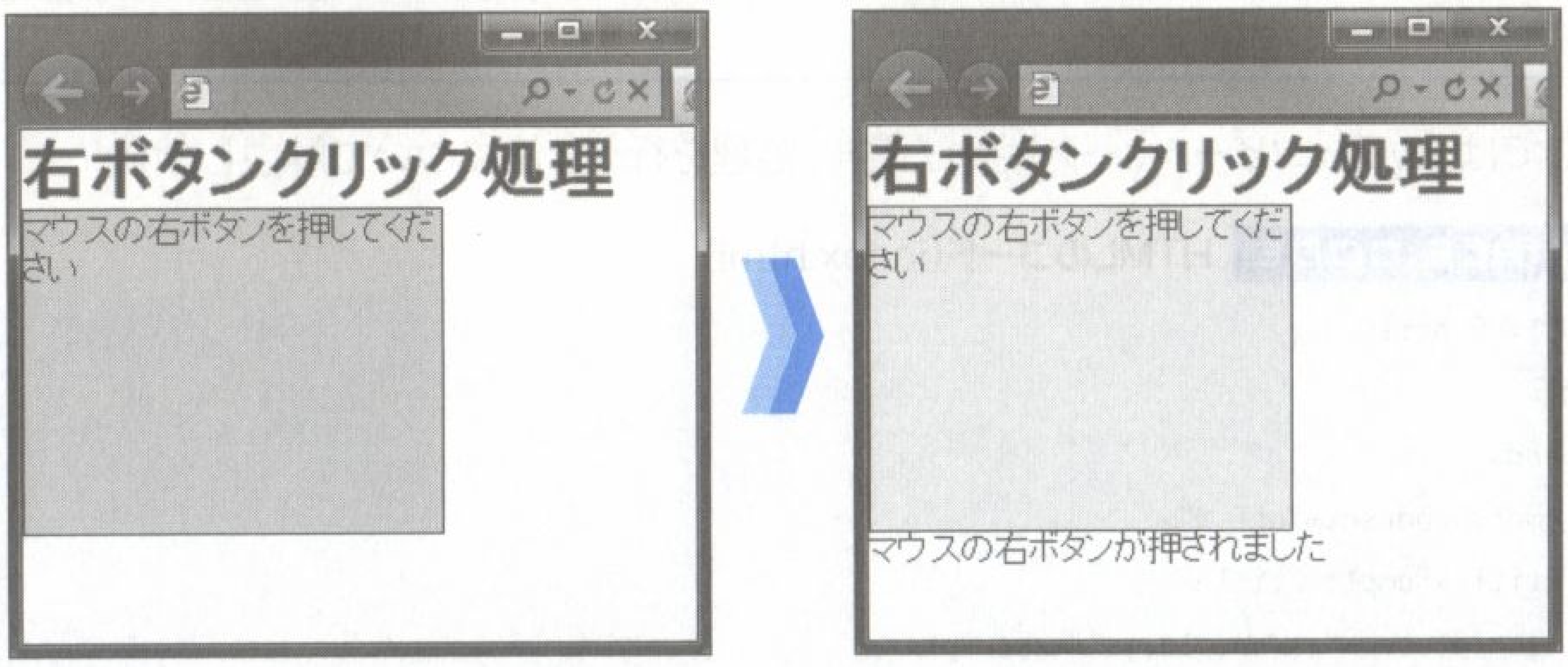
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>右ボタンクリック処理</h1>
    <div>マウスの右ボタンを押してください</div>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // div要素への参照を変数に代入
  var myArea = document.getElementsByTagName("div")[0];
  // contextmenuイベントを割り当て
  myArea.addEventListener("contextmenu", function(evt){
    // 黄色に背景色を変更
    evt.target.style.backgroundColor = "yellow";
    ele.innerHTML = "マウスの右ボタンが押されました";
    // デフォルトのイベントを禁止
    evt.preventDefault();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、画面内の四角い領域でマウスの右ボタンを押すと、マウスの右ボタンが押されたことを示すメッセージが表示され背景色が黄色に変わります。



ONEPOINT **マウスの右ボタンが押されたときに処理するには「contextmenu」イベントを捕捉する**

マウスの右ボタンが押されたときに処理するには、「contextmenu」イベントを捕捉します。イベントを割り当てる要素に対して、「oncontextmenu」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使って設定します。呼び出されるイベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトにはイベントが発生した要素の情報も入っているため、サンプルのように要素に対して処理を行うこともできます。

「contextmenu」イベント発生時に処理を行った場合、ブラウザが用意しているデフォルトのメニューが表示されます。これを防ぐには、「oncontextmenu」プロパティにイベントハンドラを割り当てた場合、処理を終わらせる際に「return false」を指定します。イベントリスナーを設定した場合は、イベントオブジェクトの「preventDefault()」メソッドを呼び出します。

COLUMN **Windows Phoneの場合**

Windows Phoneでは、要素にタッチした場合でも「contextmenu」イベントが発生します。

関連項目 ▶▶▶

● マウスダウン時/マウスアップ時の処理を行う ……………… p.418

SECTION-184

カット/コピー/ペーストされた際に処理を行う

ここでは、カット/コピー/ペーストされた際に処理を行う方法について解説します。

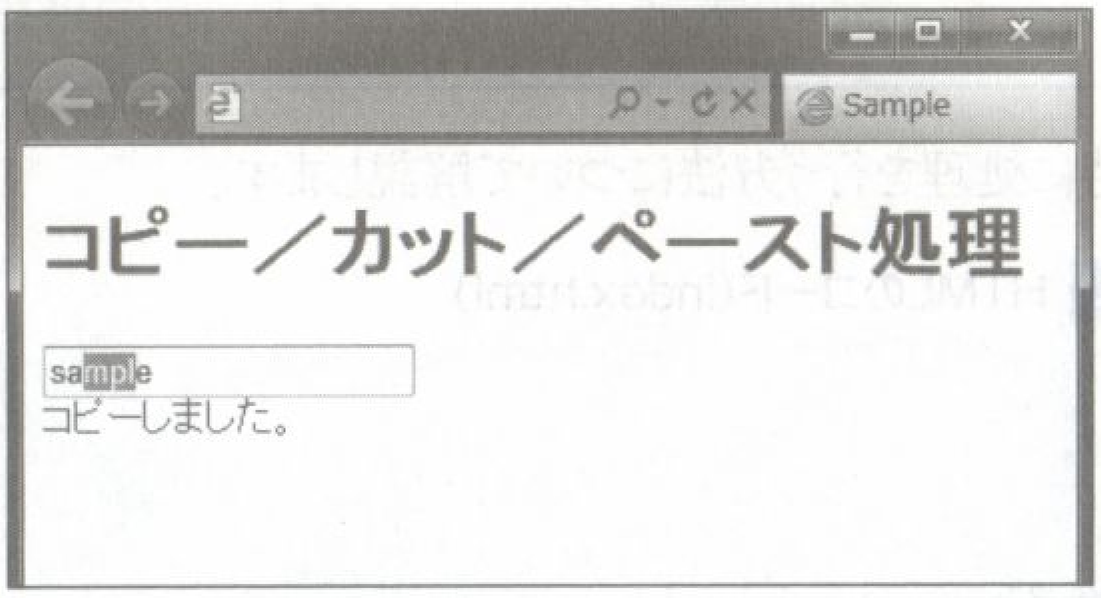
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>コピー／カット／ペースト処理</h1>
    <form>
      <input type="text" value="sample" id="myFld"><br>
    </form>
    <output></output>
  </body>
</html>
```

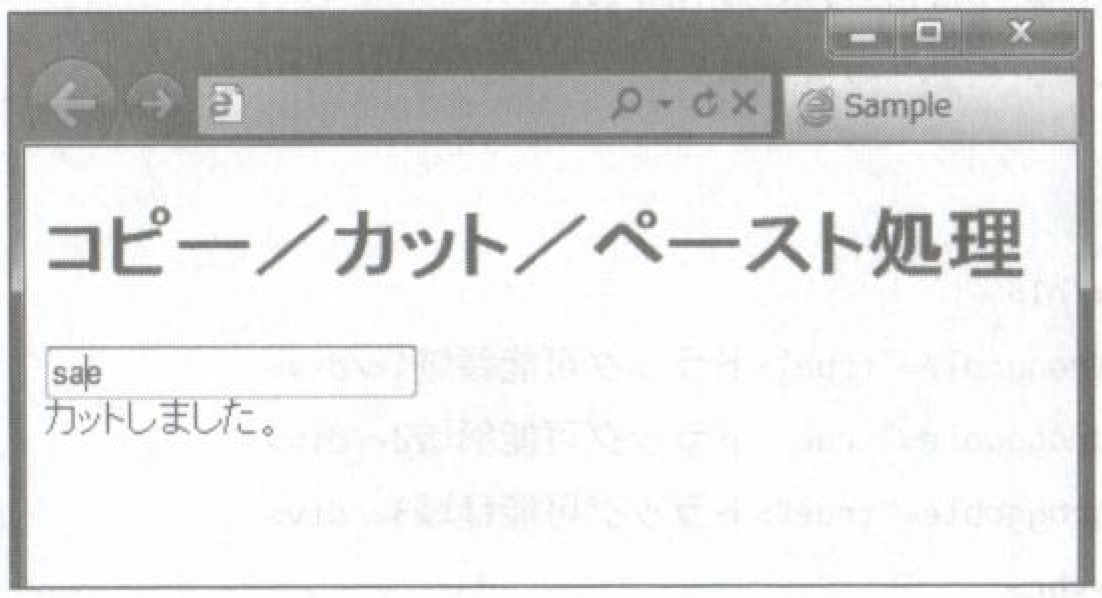
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // テキストフィールドの参照を変数に代入
  var fld = document.getElementById("myFld");
  // テキストフィールドにcopyイベントを割り当て
  fld.addEventListener("copy", function(evt){
    ele.innerHTML = "コピーしました。";
  }, false);
  // テキストフィールドにcutイベントを割り当て
  fld.addEventListener("cut", function(evt){
    ele.innerHTML = "カットしました。";
  }, false);
  // テキストフィールドにpasetイベントを割り当て
  fld.addEventListener("paste", function(evt){
    ele.innerHTML = "ペーストしました。";
  }, false);
}, false);
```

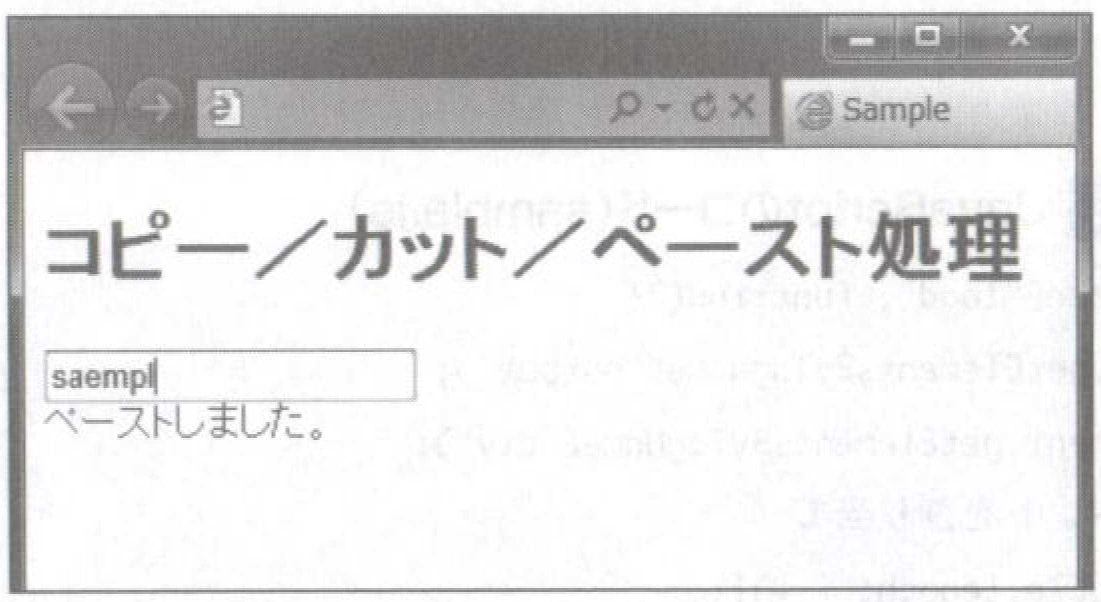
ブラウザでindex.htmlを表示し、テキストフィールドの内容をコピーすると、コピーしたことを示す文字が表示されます。



テキストフィールドの内容をカットすると、カットしたことを示す文字が表示されます。



テキストフィールドにペーストすると、ペーストしたことを示す文字が表示されます。



ONEPOINT カット・コピー・ペーストされた際に処理を行うには「copy」イベントや「paste」イベントなどを補足する

カット・コピー・ペーストされた際に処理を行うには、次に示すイベントを捕捉します。

イベント	説明
copy	コピーしたとき
beforecopy	コピーする前
cut	カットしたとき
beforecut	カットする前
paste	ペースト(貼り付け)したとき
beforepaste	ペースト(貼り付け)する前

要素では「oncopy」のように、「on」を付けたプロパティにイベントハンドラを設定できます。また、「addEventListener()」メソッドを使ってイベントリスナーで設定することもできます。

ドラッグ時に処理を行う

ここでは、ドラッグ時に処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

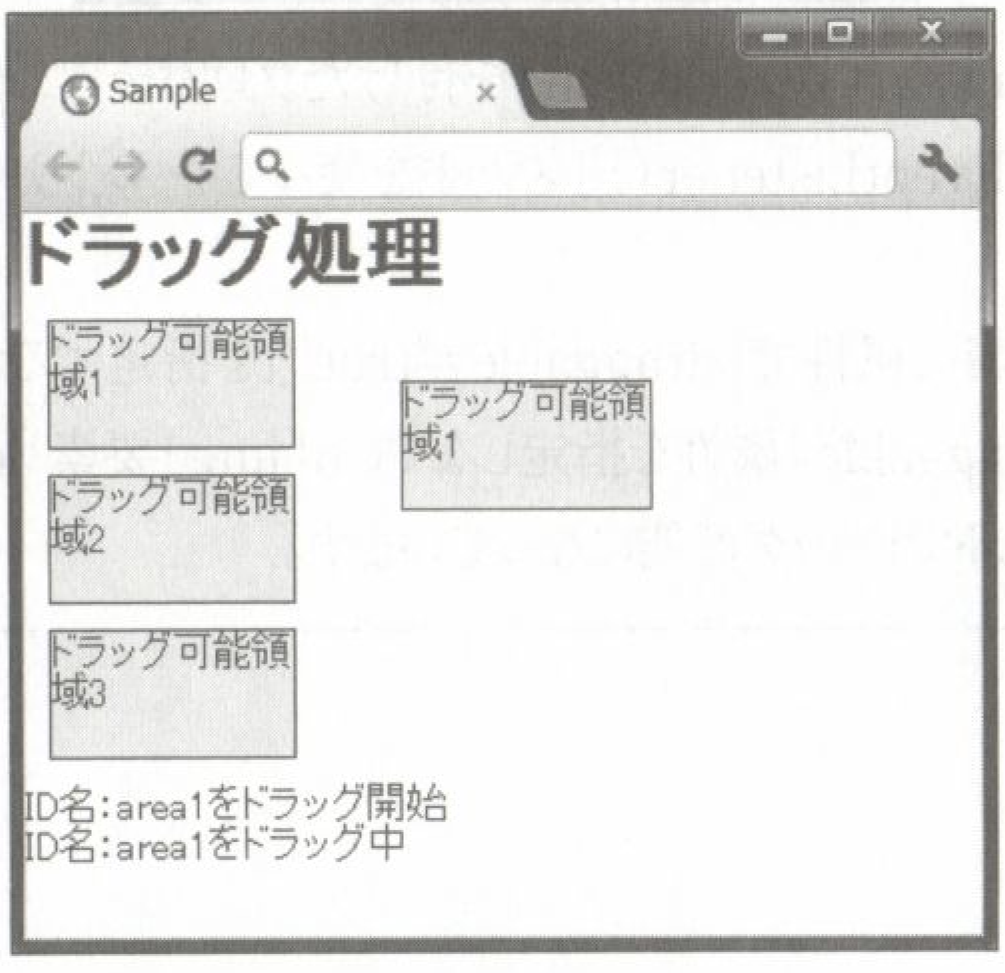
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ドラッグ処理</h1>
    <div id="area1" draggable="true">ドラッグ可能領域1</div>
    <div id="area2" draggable="true">ドラッグ可能領域2</div>
    <div id="area3" draggable="true">ドラッグ可能領域3</div>
    <output></output><br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

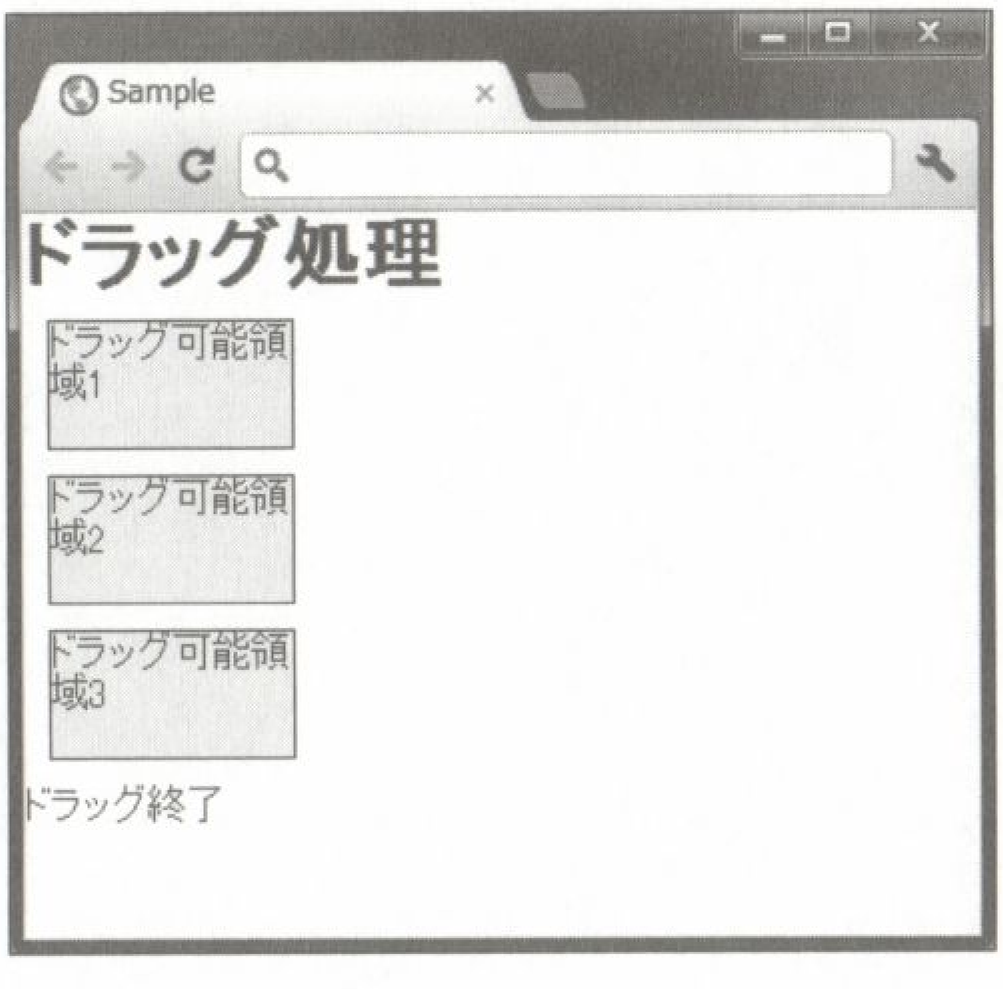
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  var dragEle = document.getElementsByTagName("div");
  // ドラッグ関連イベントを割り当て
  for(var i=0; i<dragEle.length; i++){
    // ドラッグ開始イベントを設定
    dragEle[i].addEventListener("dragstart", function(evt){
      var id = evt.target.id;
      ele[0].innerHTML = "ID名："+id+"をドラッグ開始";
    }, false);
    // ドラッグ中イベントを設定
    dragEle[i].addEventListener("drag", function(evt){
      var id = evt.target.id;
      var time = (new Date()).getTime();
      ele[1].innerHTML = "ID名："+id+"をドラッグ中";
    }, false);
    // ドラッグ開始イベントを設定
    dragEle[i].addEventListener("dragend", function(evt){
      var id = evt.target.id;
      ele[0].innerHTML = "ドラッグ終了";
```

```
        ele[1].innerHTML = "";
    }, false);
  }
}, false);
```

ブラウザでindex.htmlを表示し、「div」要素をドラッグすると、ドラッグ開始のメッセージとIDが表示されます。ドラッグ中にもメッセージが表示されます。



ドラッグが終了すると、終了のメッセージが表示されます。



ONEPOINT

ドラッグ時に処理を行うには「dragstart」「drag」「dragend」イベントを補足する

ドラッグ時に処理を行うには、次に示すイベントを捕捉します。

イベント	説明
dragstart	ドラッグを開始したとき
drag	ドラッグ中
dragend	ドラッグを終了したとき

要素では、「ondragstart」のように「on」を付けたプロパティにイベントハンドラを設定できます。また、「addEventListener()」メソッドを使ってイベントリスナーを使って設定することもできます。

HTML5では、要素に属性で「draggable="true"」を指定するだけでドラッグが可能になります。また、「draggable」属性を指定しなくても「img」要素（画像）や「a」要素、選択した文字はデフォルトでドラッグ可能になっています。

SECTION-186

ページの読み込みやDOMの構築が完了した際に処理を行う

ここでは、DOMの構築が完了した際に処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>DOMContentLoaded/loadイベント</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // loadイベントが発生した事を画面に出力
  ele.innerHTML += "loadイベント発生<br>";
}, false);
window.addEventListener("DOMContentLoaded", function(){
  var ele = document.getElementsByTagName("output")[0];
  // DOMContentLoadedイベントが発生した事を画面に出力
  ele.innerHTML += "DOMContentLoadedイベント発生<br>";
}, false);
```

ブラウザでindex.htmlを表示すると、イベントが完了した際にメッセージが順番に表示されます。



ONEPOINT

DOMの構築が完了した際に処理を行うには「DOMContentLoaded」イベントを検知する

DOMの構築が完了した際に処理を行うには、「DOMContentLoaded」イベントを検知します。「DOMContentLoaded」イベントは、ページの読み込みが完了したことを示す「load」イベントよりも先に発生します。「DOMContentLoaded」はページ上のDOMの使用が可能になったことを示すので、このイベントが発生した後であれば、スクリプトでDOM操作を行うことができます。

SECTION-187

リセット時・送信時に処理する

ここでは、送信時・リセット時に処理する方法について解説します。

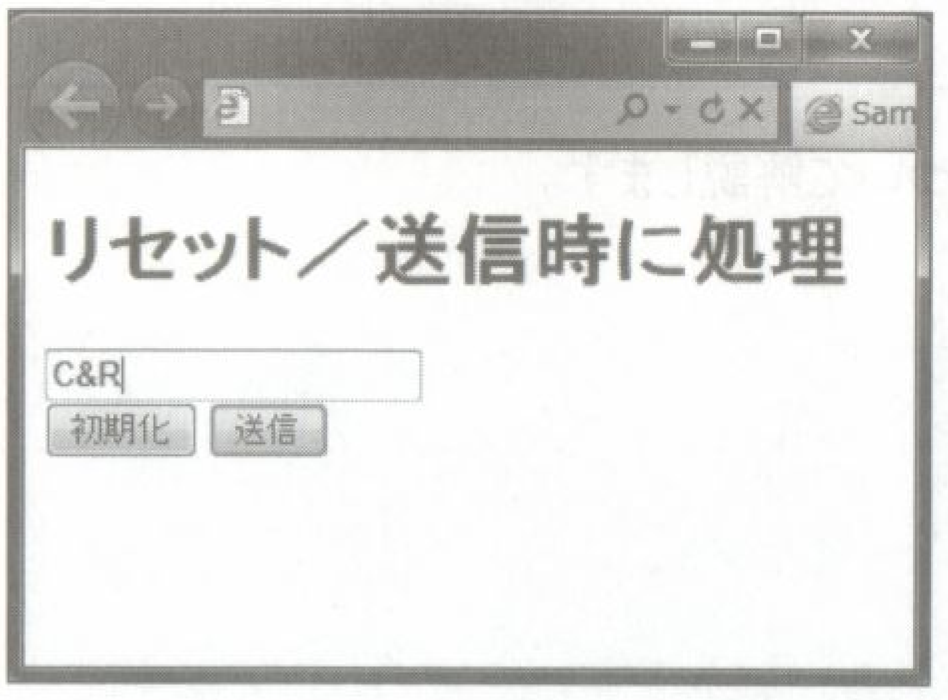
SAMPLE CODE HTMLのコード(index.html)

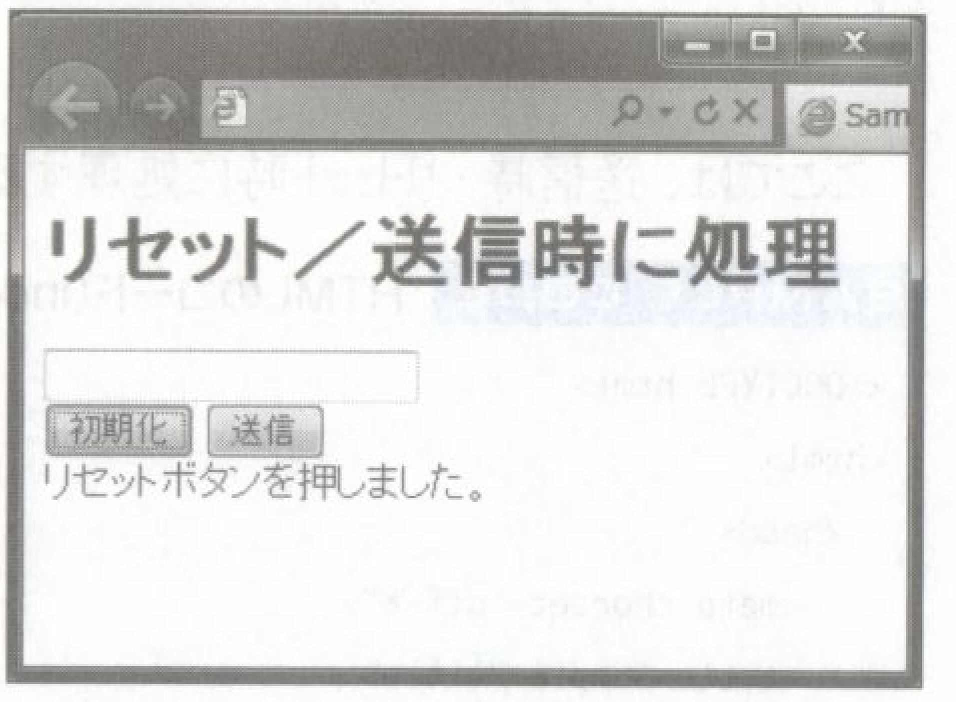
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>リセット／送信時に処理</h1>
    <form>
      <input type="text" id="myFld"><br>
      <input type="reset" value="初期化" id="rstBtn">
      <input type="submit" value="送信" id="sndBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

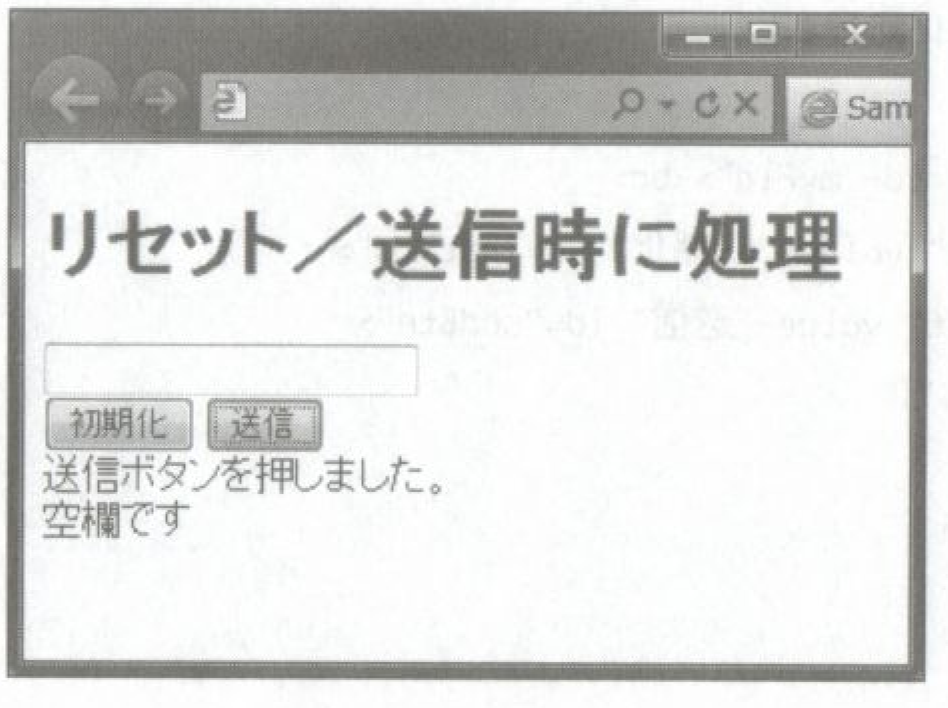
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // フォームの参照を変数に代入
  var myForm = document.getElementsByTagName("form")[0];
  // フォームにresetイベントを割り当て
  myForm.addEventListener("reset", function(evt){
    ele.innerHTML = "リセットボタンを押しました。";
  }, false);
  // テキストフィールドにcutイベントを割り当て
  myForm.addEventListener("submit", function(evt){
    ele.innerHTML = "送信ボタンを押しました。";
    // 空欄チェック
    var text = document.getElementById("myFld").value;
    if (!text){
      ele.innerHTML += "<br>空欄です";
      // デフォルトのイベントを禁止
      evt.preventDefault();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに文字を入力した後に初期化ボタンをクリックすると、テキストフィールドの内容が初期化(この場合は何も文字がない状態)されます。





送信ボタンをクリックすると空欄チェックが行われ、空欄の場合はその旨メッセージが表示されます。



ONEPOINT	リセット時に処理するには「reset」イベント、送信時に処理するには「submit」イベントを補足する

リセットボタンがクリックされると、「form」要素で「reset」イベントが発生します。「form」要素の「onreset」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントハンドラを設定します。

送信ボタンがクリックされると、「form」要素で「submit」イベントが発生します。「form」要素の「onsubmit」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントハンドラを設定します。空欄チェックなどを行った際に、データを送信したくない場合があります。データを送信したくない場合、「onsubmit」プロパティにイベントハンドラを設定したら、最後に「return false」とします。イベントリスナーの場合は、イベントオブジェクトの「preventDefault()」メソッドを呼び出します。これによって、フォームに入力されたデータは送信されなくなります。

関連項目 ▶▶▶

● フォーム内容を初期化する …………………………………………………… p.333

SECTION-188

スクロール時に処理を行う

ここでは、スクロール時に処理を行う方法について解説します。

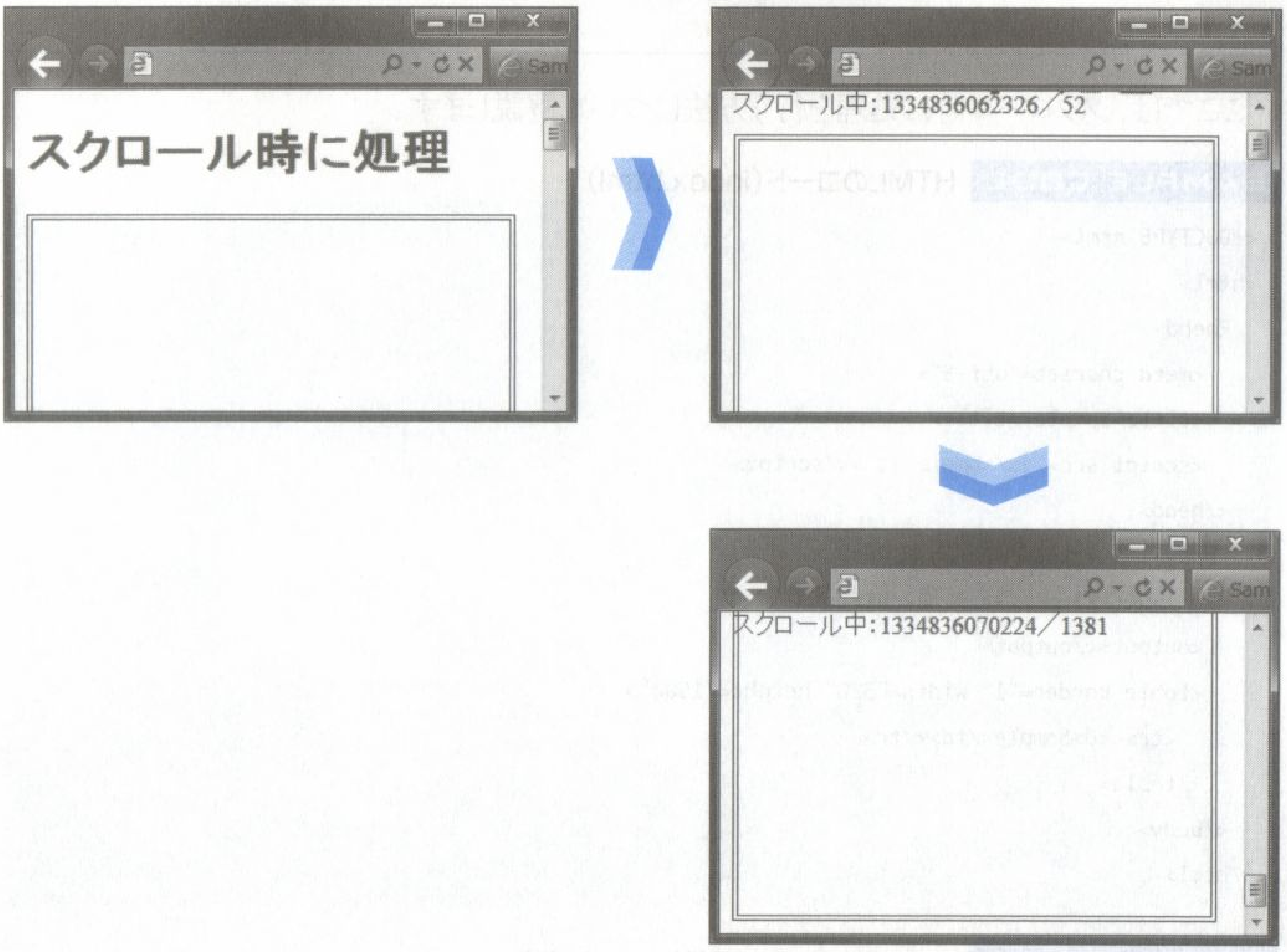
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>スクロール時に処理</h1>
    <output></output>
    <table border="1" width="320" height="1500">
      <tr><td>Sample</td></tr>
    </table>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // body要素のスクロールイベントを設定
  window.addEventListener("scroll", function(evt){
    var time = (new Date()).getTime();
    ele.innerHTML = "スクロール中 :"+time;
    // スクロール位置を読み出し
    var posY = document.documentElement.scrollTop || document.body.scrollTop;
    ele.innerHTML += "／"+posY;
    // output要素の表示位置を設定
    ele.style.position = "absolute";
    ele.style.top = posY + "px";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、スクロールすると、時間を示すミリ秒とスクロール量が表示されます。



ONEPOINT スクロール時に処理するには「scroll」イベントを捕捉する

スクロールすると、要素の対して「scroll」イベントが発生します。イベントが発生したら処理するには、要素の「onscroll」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントハンドラを設定します。

スクロール位置を求める場合は、次のプロパティを参照します。

プロパティ	説明
document.documentElement.scrollTop、または、document.body.scrollTop	上からのスクロール量
document.documentElement.scrollLeft、または、document.body.scrollLeft	左からのスクロール量

なお、iPhoneやiPad、Androidの場合、scrollイベントはパソコンとは異なり、スクロールさせた後に発生します。

関連項目 ▶▶▶

● 指定位置までスクロールさせる ……………………………… p.269

SECTION-189

テキストフィールド内のテキスト選択時に処理を行う

ここでは、テキストフィールド内のテキスト選択時に処理を行う方法について解説します。

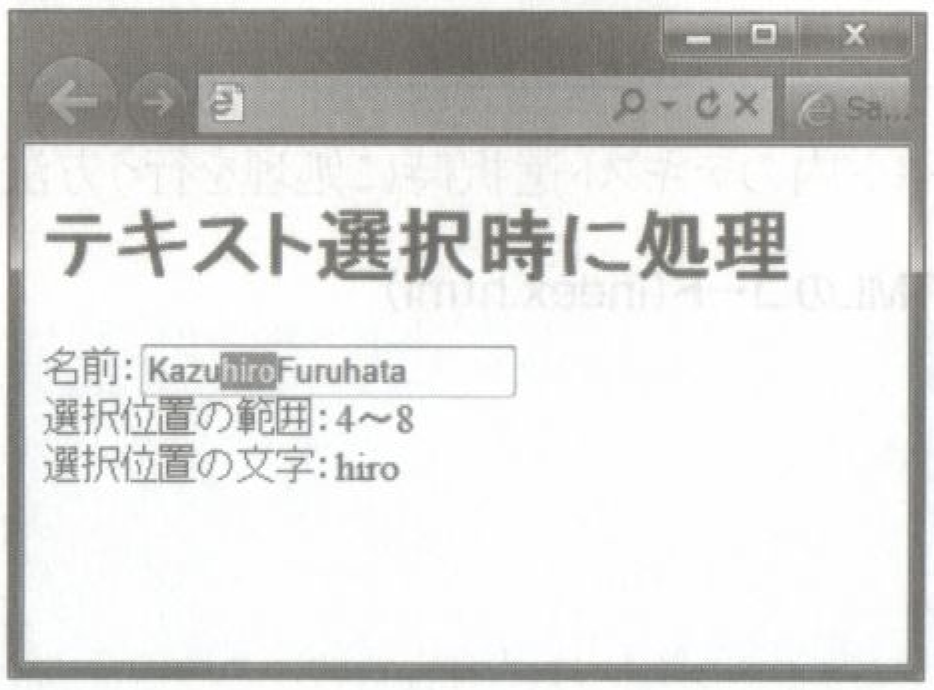
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>テキスト選択時に処理</h1>
    <form>
      名前：<input type="text" value="KazuhiroFuruhata" id="uName"><br>
    </form>
    <div>選択位置の範囲：<output></output></div>
    <div>選択位置の文字：<output></output></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  // テキストフィールドにイベントを設定
  document.getElementById("uName").addEventListener("select", function(evt){
    // 選択開始位置を取得
    var startPos = this.selectionStart;
    // 選択終了位置を取得
    var endPos = this.selectionEnd;
    // 選択範囲を出力
    ele[0].innerHTML = startPos+"～"+endPos;
    // 選択範囲の文字を抜き出す
    var text = this.value.substring(startPos, endPos);
    // 結果を出力
    ele[1].innerHTML = text;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールド内のテキストが選択されると、選択範囲の位置を選択した文字が表示されます。



ONEPOINT　テキスト選択時に処理を行うには「select」イベントを捕捉する

テキストフィールド内のテキストが選択されると、「select」イベントが発生します。テキストフィールドの「onselect」にイベントハンドラを設定するか、「addEvenetListener()」メソッドを使ってイベントハンドラを設定します。

テキストフィールド内の文字が選択された際、選択された文字の範囲や文字そのものを取得することができます。イベントハンドラ内で「this.selectionStart」プロパティで選択開始位置、「this.selectionEnd」プロパティで選択終了位置を取得することができます。選択された文字は「this.value」に文字列全体が入っているので、「substring()」メソッドを使って抜き出します。

COLUMN　IE6〜8で動作させるには

サンプルはIE9以降では動作しますが、IE6〜8ではエラーとなって動作しません。IE6〜8でも動作させる場合は、次のコードのようになります。

SAMPLE CODE　HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>テキスト選択時に処理</h1>
    <form>
      名前：<input type="text" value="KazuhiroFuruhata" id="uName"><br>
    </form>
    <div>選択位置の範囲：<span></span></div>
```

```
    <div>選択位置の文字：<span></span></div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.onload = function(){
  var ele = document.getElementsByTagName("span");
  // テキストフィールドにイベントを設定
  document.getElementById("uName").onselect = function(evt){
    var d = document.selection.createRange();
    var te = this.createTextRange();
    te.setEndPoint("EndToStart", d);
    var startPos = te.text.length;
    te.setEndPoint("EndToEnd", d);
    var endPos = te.text.length;
    ele[0].innerHTML = startPos+"～"+endPos;
    var text = this.value.substring(startPos, endPos);
    ele[1].innerHTML = text;
  }
};
```

SECTION-190

指定した座標にある複数の要素を求める

IE10では指定した座標にある要素を求める機能が拡張されました。ここでは、指定した座標にある要素を求める方法について解説します。

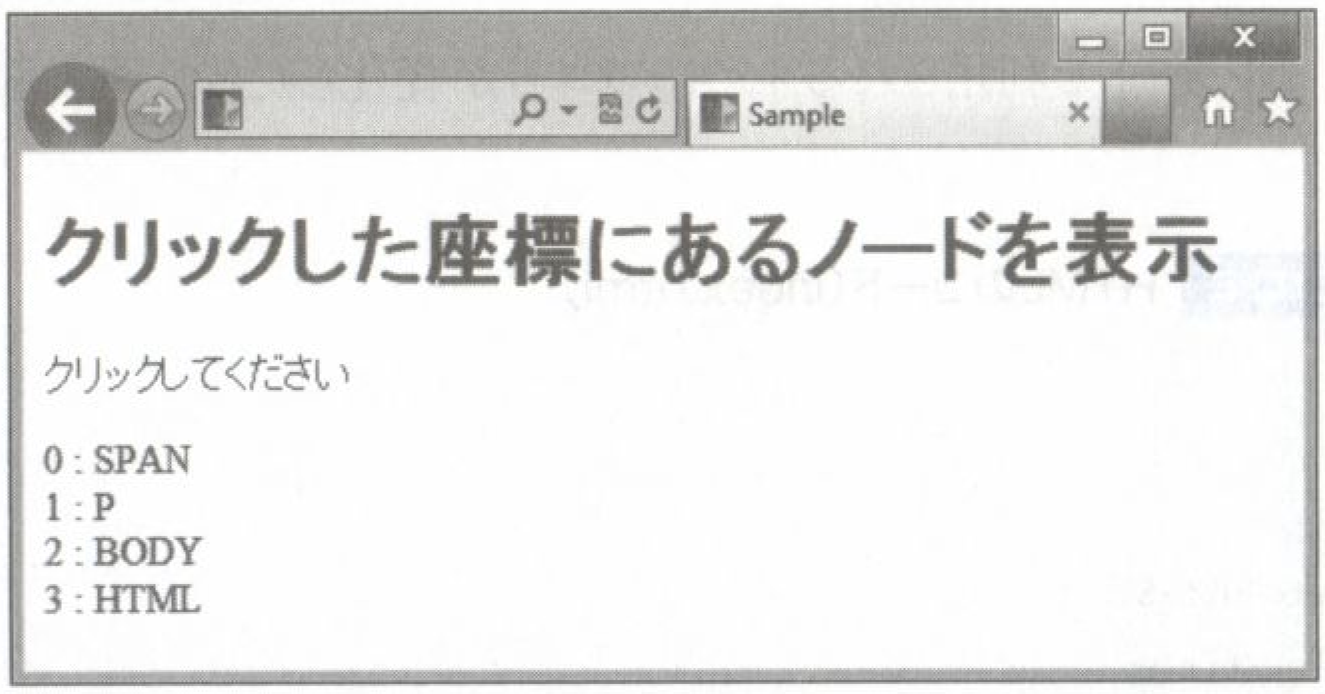
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>クリックした座標にあるノードを表示</h1>
    <p><span>クリック</span>してください</p>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // body要素にイベントリスナーを設定
  window.addEventListener("click", function(evt){
    // 表示する領域を消去
    ele.innerHTML = "";
    // クリックした座標を取得
    var cx = evt.clientX;
    var cy = evt.clientY;
    // クリック位置のノードリストを取得
    var nodeList = document.msElementsFromPoint(cx, cy);
    // ノードリストのタグ名一覧を出力
    for(var i=0; i<nodeList.length; i++){
      ele.innerHTML += i+" : "+nodeList[i].tagName+"<br>";
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ページ内の任意の場所をクリックすると、クリックした場所にあるノードの一覧が表示されます。ノードのタグ名は、HTMLの記述順で内側のものから外側のものに向かって表示されます。



ONEPOINT　指定した座標にある要素を求めるには「elementFromPoint()」「elementsFromPoint()」「elementsFromRect()」メソッドを使う

指定した座標にある要素を求めるには、「elementFromPoint()」メソッド、「elementsFromPoint()」メソッド、「elementsFromRect()」メソッドのいずれかを使います。「elementFromPoint()」メソッドはIE6以降で実装されており、パラメータに指定したx座標・y座標上にある要素を返します。ただし、要素が重なっている場合には、最も手前の要素しか返されません。

重なっている要素すべてを取得する場合は、「elementsFromPoint()」メソッドを使います。このメソッドは、指定した座標にある要素すべてをノードリストとして返します。また、範囲を指定して要素を取得する場合は、「elementsFromRect()」メソッドを使います。「elementsFromRect()」メソッドのパラメータは、左座標、上座標、横幅、縦幅になります。

なお、2012年7月時点では、「elementsFromPoint()」メソッドと「elementsFromRect()」メソッドには、ベンダープレフィックスが必要です（サンプルでも「ms」のベンダープレフィックスを付けている）。

SECTION-191

CSSのメディアクエリが変化したら処理する

ここでは、スタイルシートで利用されるメディアクエリが変化したら処理する方法について解説します。

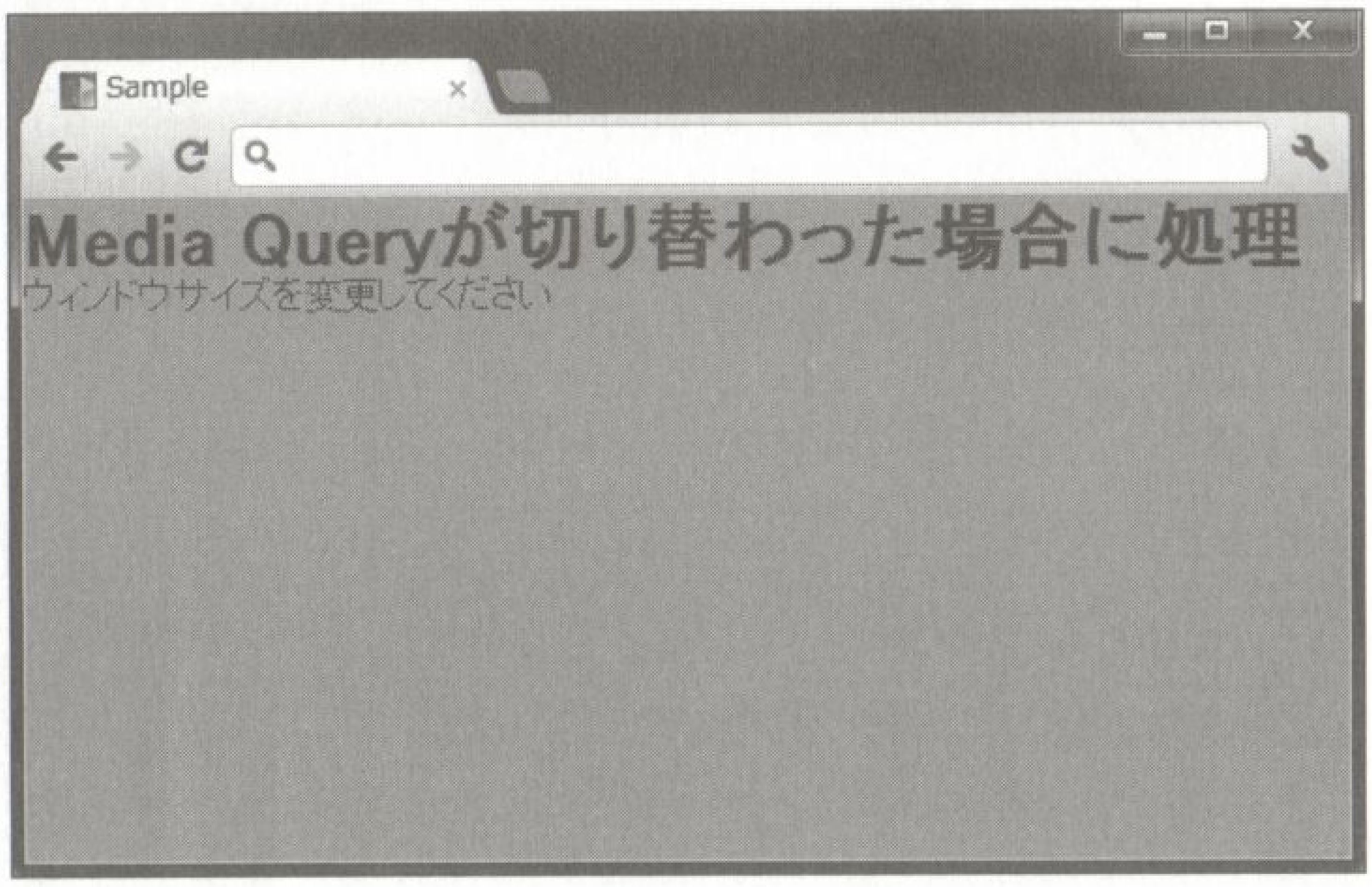
SAMPLE CODE HTMLのコード(index.html)

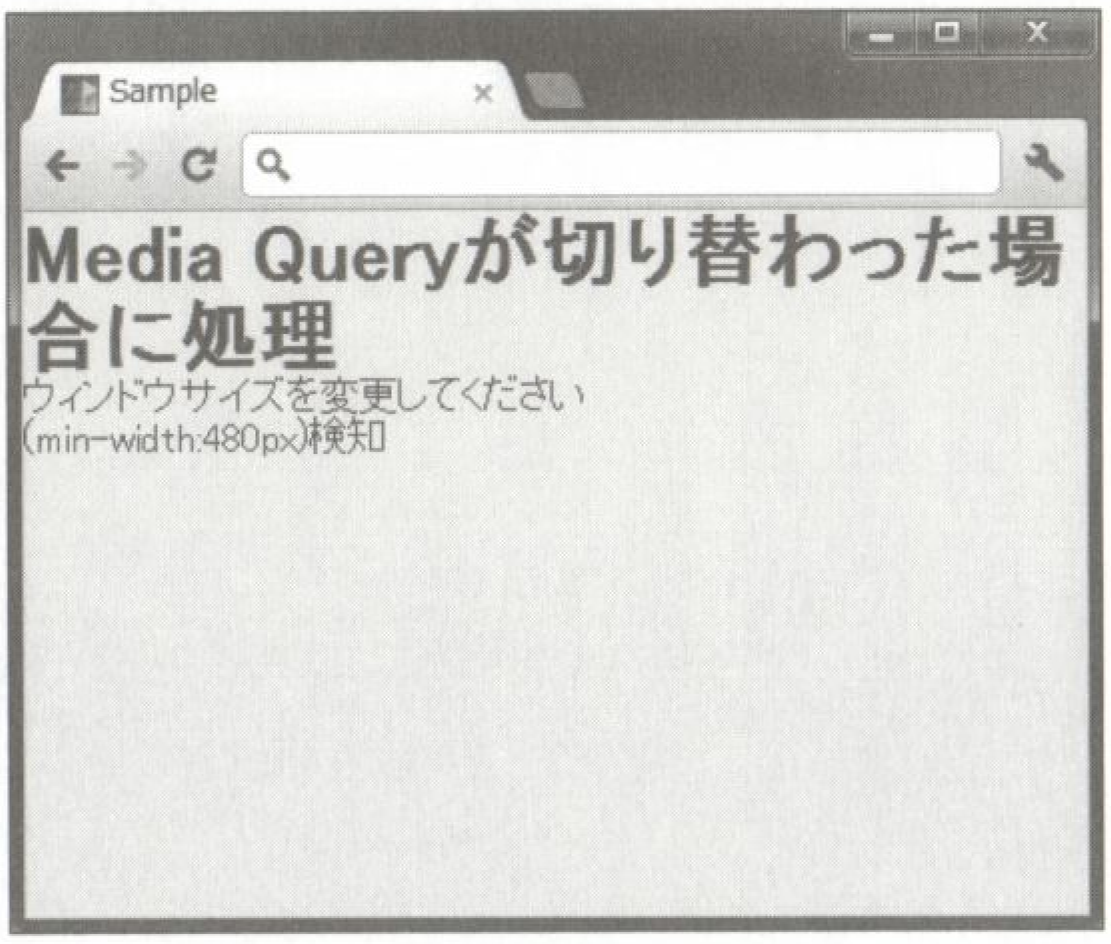
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" type="text/css" href="css/min.css"
      media="screen and (min-width:240px) and (max-width:480px)">
    <link rel="stylesheet" type="text/css" href="css/max.css"
      media="screen and (min-width:480px) and (max-width:1024px)">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>Media Queryが切り替わった場合に処理</h1>
    <p>ウィンドウサイズを変更してください</p>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // マッチするMediaを指定。IE10はmsMatchMedia
  window.matchMedia("(min-width:480px)").addListener(function(){
    var ele = document.getElementsByTagName("output")[0];
    ele.innerHTML += "(min-width:480px)検知<br>";
  });
}, false);
```

ブラウザでindex.htmlを表示します。ウィンドウの横幅に応じてページの背景色が変化するCSSファイルが読み込まれており、ウィンドウ幅を変化させると、該当するメディアクエリとマッチした場合メッセージが表示されます。





ONEPOINT **メディアクエリが変化したら処理するには「window.matchMedia()」メソッドと「addListener()」メソッドを使う**

メディアクエリが変化したら処理するには、「window.matchMedia()」メソッドと「addListener()」メソッドを使います。パラメータにはマッチさせるクエリ文字列を指定します。また、メディアクエリが変化した場合に処理するために、「addListener()」メソッドを使います。「addListener()」メソッドのパラメータには、メディアクエリが変化した際に呼び出される関数を指定します。

なお、IE10では、「matchMedia()」メソッドを使用するには、「msMatchMedia()」のようにベンダープレフィックスを付ける必要があります。

関連項目 ▶▶▶

● CSSのメディアクエリを調べる …… p.973

CHAPTER 10

Canvas/ Canvas 3D/SVG

SECTION-192

コンテキストを取得する

ここでは、Canvasの描画を扱う場合に必要なコンテキストを取得する方法について解説します。

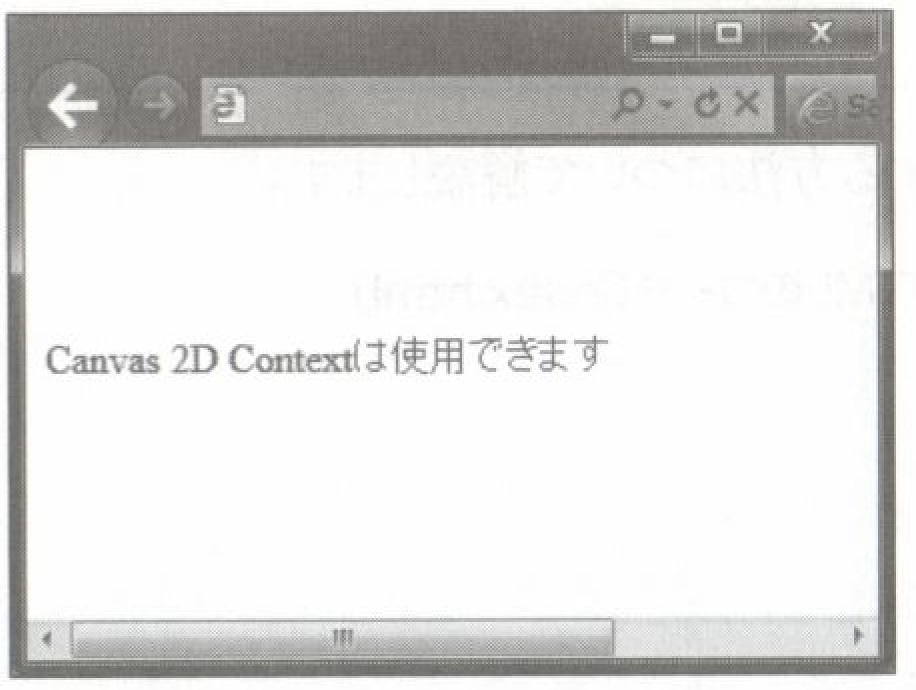
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="480" height="64">
      Canvas対応ブラウザを使用して下さい
    </canvas>
    <br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (context == null){
    ele.innerHTML = "Canvas 2D Contextは使用できません";
  }else{
    ele.innerHTML = "Canvas 2D Contextは使用できます";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、Canvas 2Dコンテキストが使用可能かどうか調べた結果が表示されます。



ONEPOINT **コンテキストを取得するには「getContext()」メソッドを使う**

Canvasには、ピクセル単位で処理を行う二次元画像を扱う2D Contextと、3D描画を扱うWebGL用のコンテキストがあります。他にもブラウザ独自にコンテキストが用意されている場合もあります。

このコンテキストを取得するには、「getContext()」メソッドを使います。パラメータには、取得したいコンテキスト名を指定します。2Dの場合は「"2d"」を、3Dの場合は「"web-gl"」(ベンダープレフィックスが必要な場合もある)を指定します。

コンテキストが取得できなかった場合、「getContext()」メソッドは、「null」を返します。また、「getContext()」を使った場合、後から他のコンテキストを取得することはできません。つまり、Canvasには1つのコンテキストしか利用できないことになります(2Dと3Dの混在はできない)。

SECTION-193

パスを作成する

ここでは、パスを作成する方法について解説します。

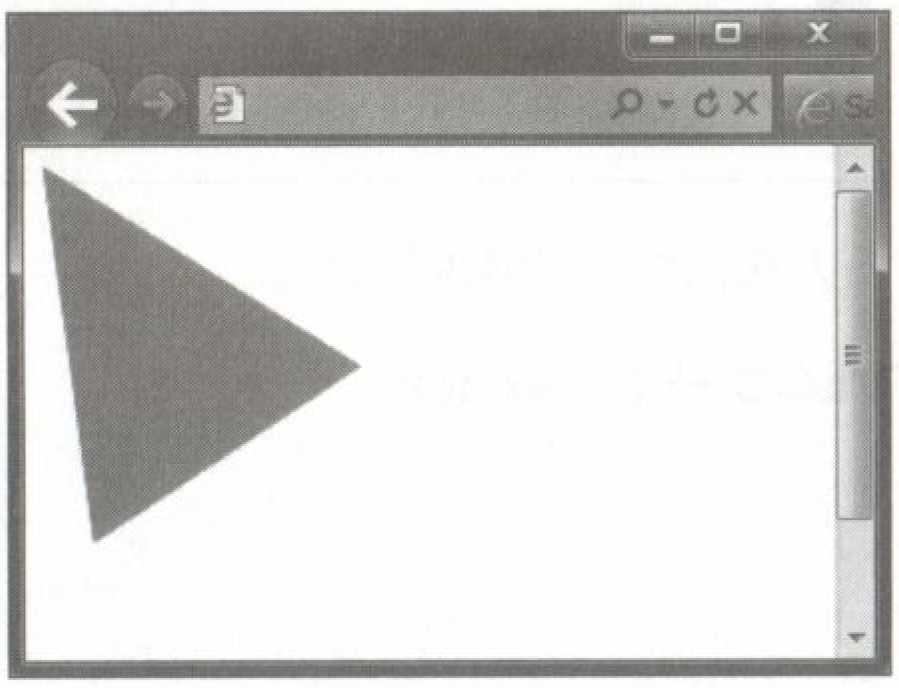
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 新規にパスを作成
  context.beginPath();
  // 図形の座標値を設定
  context.moveTo(0,0);
  context.lineTo(130, 80);
  context.lineTo(20, 150);
  // パスを閉じる(クローズパス)
  context.closePath();
  // 塗りつぶす色を設定
  context.fillStyle = "red";
  context.fill();
}, false);
```

ブラウザでindex.htmlを表示すると、パスを使って描画した三角形が表示されます。



ONEPOINT 新規にパスを作成するには「beginPath()」メソッド、クローズパスに設定するには「closePath()」メソッドを使う

Canvas 2D Contextでは、図形を描画する際、直接、描くタイプのメソッドと、パスを作成した後で描画するタイプがあります。直接、描くタイプとしては、「fillRect()」メソッドなどがあります。直接、描くタイプは、四角形などシンプルなものに限定されています。より複雑な図形を描画する場合はパスを作成し、その中で描画する図形の座標を設定していきます。座標の設定が終わったら、「stroke()」メソッドや「fill()」メソッドを使って描画します。

パスを作成するには、「beginPath()」メソッドを使います。これによって、これまで作成したパスは破棄され、新規にパスが作成されます。パスの作成は、「lineTo()」メソッドや「arc()」メソッド、「rect()」メソッドなど、直接、描画しないメソッドを利用して行います。作成したパスは、オープンパスとクローズパスの2種類になります。始点と終点を結ばない場合がオープンパス、始点と終点を結ぶ場合はクローズパスになります。クローズパスにする場合は、「closePath()」メソッドを使います。

なお、作成するパスはベクトルの向き(描画方向)が時計回りか反時計回りかによって、パスが重なった部分の表示結果が異なります。

COLUMN Androidでの不具合

Android 2.0〜4.0の標準ブラウザでは、「arc()」メソッドでの時計回り/反時計回りの指定は無視されます。このため、ベクトルの向きによってくり抜く(ドーナツ型など)ことはできません。Android 4.1以降の標準ブラウザ(Chrome for Android)やFirefoxなど、他のブラウザなら正しく処理されます。

SECTION-194

直線を描画する

ここでは、直線を描画する方法について解説します。

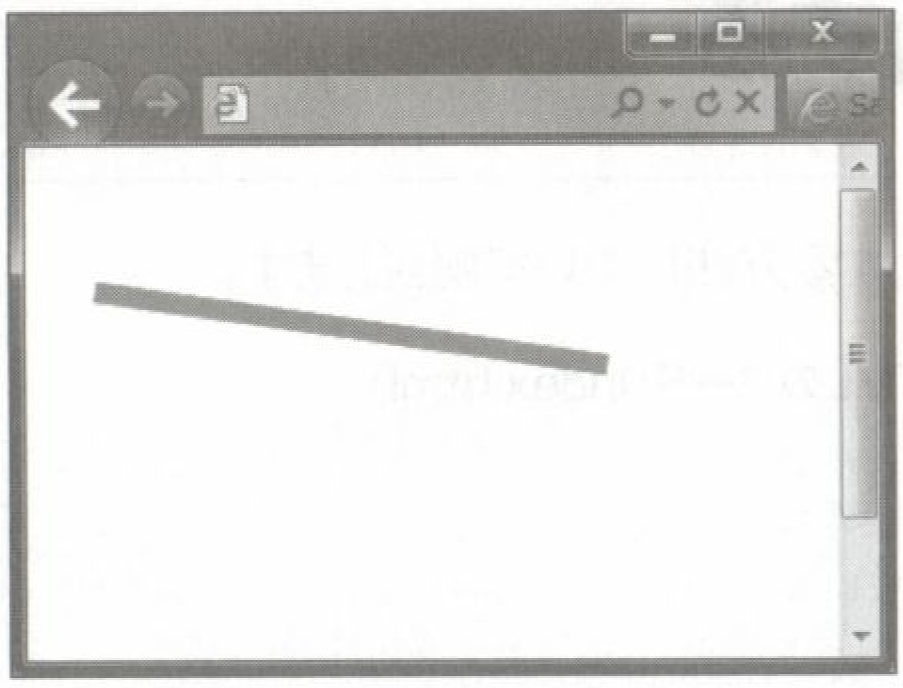
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 新規にパスを作成
  context.beginPath();
  // 直線の開始座標を指定
  context.moveTo(20, 50);
  context.lineTo(230, 80);
  // 線の色とサイズを設定
  context.strokeStyle = "red";
  context.lineWidth = 8;
  // 直線を描画
  context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、赤色の直線が表示されます。



ONEPOINT **直線の描画を行うには「lineTo()」メソッドを使う**

直線を描画するには、「lineTo()」メソッドを使います。パラメータには、直線の終了点となるX座標とY座標を指定します。「lineTo()」は、現在のペン位置を始点にして直線を構成します。ペン位置を変更するには、「moveTo()」メソッドを使います。パラメータには、移動先のX座標とY座標を指定します。簡単な直線を描画する場合には、「moveTo()」メソッドと「lineTo()」メソッドを組み合わせます。

「lineTo()」はパスの座標を設定していくだけのメソッドなので、実際に描画する場合には、「stroke()」メソッド、または、「fill()」メソッドを使います。

COLUMN **連続して直線を描くには**

連続して直線を描くには、「lineTo()」を列記します。たとえば、次のように記述します。

```
context.moveTo(0, 0);
context.lineTo(100, 50);
context.lineTo(120, 90);
context.lineTo(30, 120);
```

四角形を描画する

ここでは、四角形を描画する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

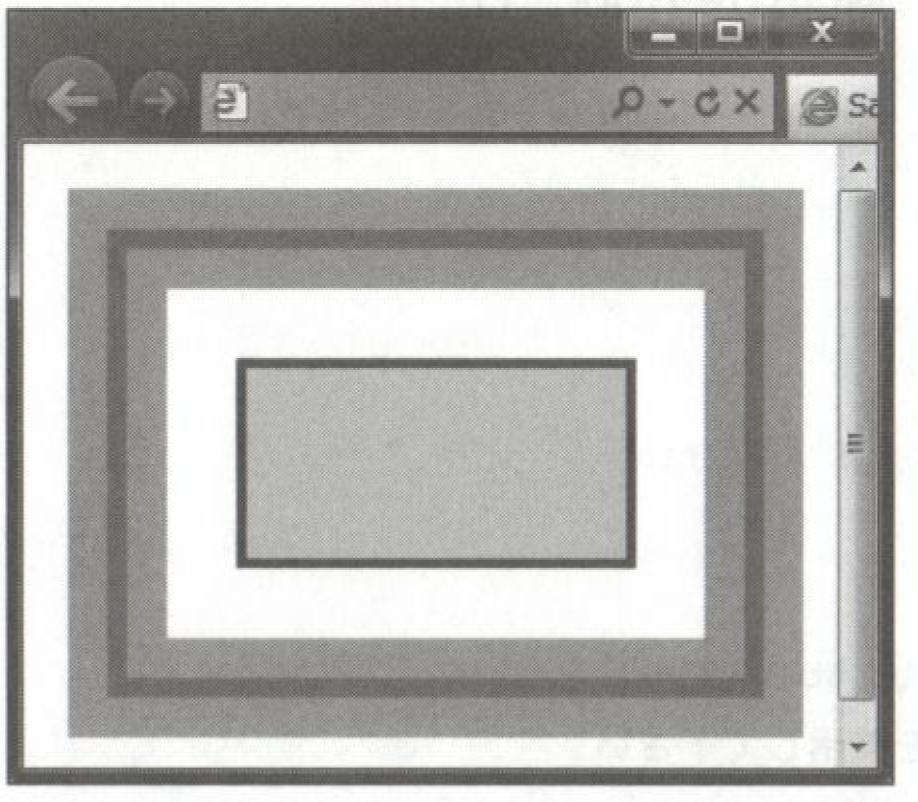
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色とサイズ、塗りの色を設定
  context.strokeStyle = "red";
  context.lineWidth = 8;
  context.fillStyle = "orange";
  // 塗り潰された四角形を描く
  context.fillRect(10,10, 300, 220);
  // 四角形で消去
  context.clearRect(50,50, 220, 140);
  // 枠だけの四角形を描く
  context.strokeRect(30,30, 260, 180);
  // 線の色とサイズ、塗りの色を設定
  context.strokeStyle = "blue";
  context.lineWidth = 4;
  context.fillStyle = "rgba(0,255,0, 0.5)";
  // パスで四角形を描く
  context.beginPath();
  context.rect(80,80, 160,80);
```

```
    context.closePath();
    context.fill();
    context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、さまざまな四角形が描画されます。



ONEPOINT

四角形の描画を行うには
「fillRect()」「strokeRect()」「clearRect()」「rect()」メソッドを使う

四角形を描画するメソッドは、次の4種類があり、用途によって異なります。

メソッド	説明
fillRect(x,y,w,h)	塗りつぶした四角形を描く
strokeRect(x,y,w,h)	枠だけの四角形を描く
clearRect(x,y,w,h)	四角形で消す(Canvasの背景色、または、透明になる)
rect(x,y,w,h)	四角形のパスを作成する

単純に四角形を描くなら、「fillRect()」メソッドか「strokeRect()」メソッドを使います。他のパスを組み合わせて複雑な図形を描く場合には、「rect()」メソッドを使います。「rect()」メソッドの場合、パスで図形を構築するため、「fill()」メソッドか、「stroke()」メソッドを使わないと描画されません。

SECTION-196

円や円弧を描く

ここでは、円や円弧を描く方法について解説します。

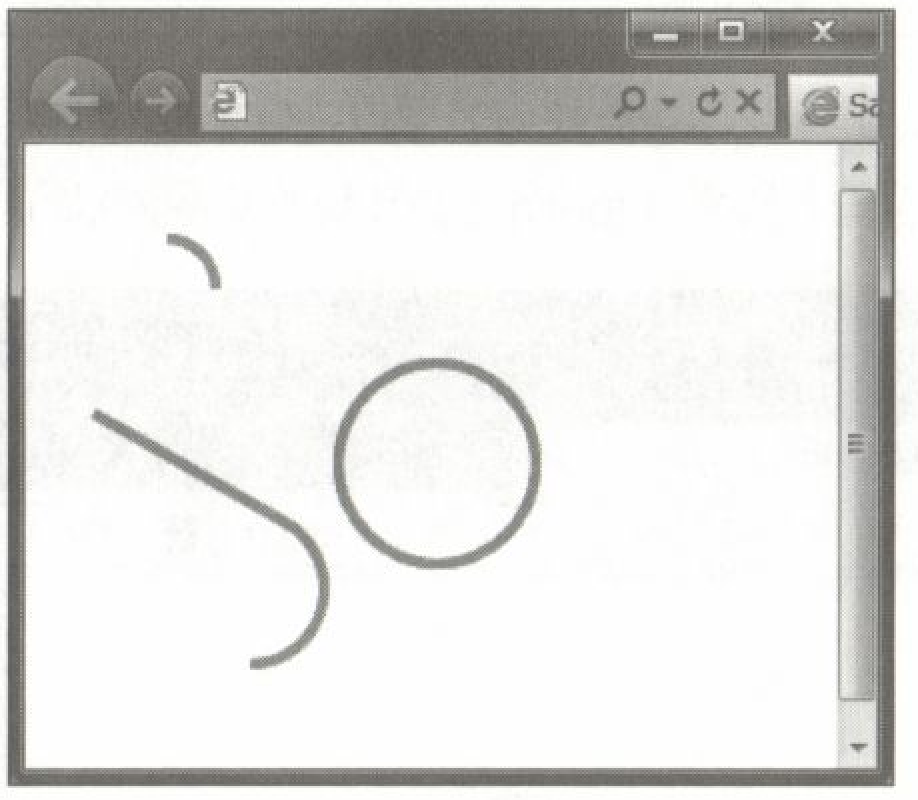
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色とサイズを設定
  context.strokeStyle = "red";
  context.lineWidth = 4;
  // 正円を描く
  context.beginPath();
  context.arc(160,120, 40, 0, Math.PI*2, true);
  context.stroke();
  // 円弧を描く
  context.beginPath();
  context.arc(50,50, 20, 0, -Math.PI/2, true);
  context.stroke();
  // 円弧を描く(2)
  context.beginPath();
  context.moveTo(20,100);
  context.arcTo(200,200, 150, 200, 30);
  context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、正円、円弧、直線と円弧の3種類が描画されます。



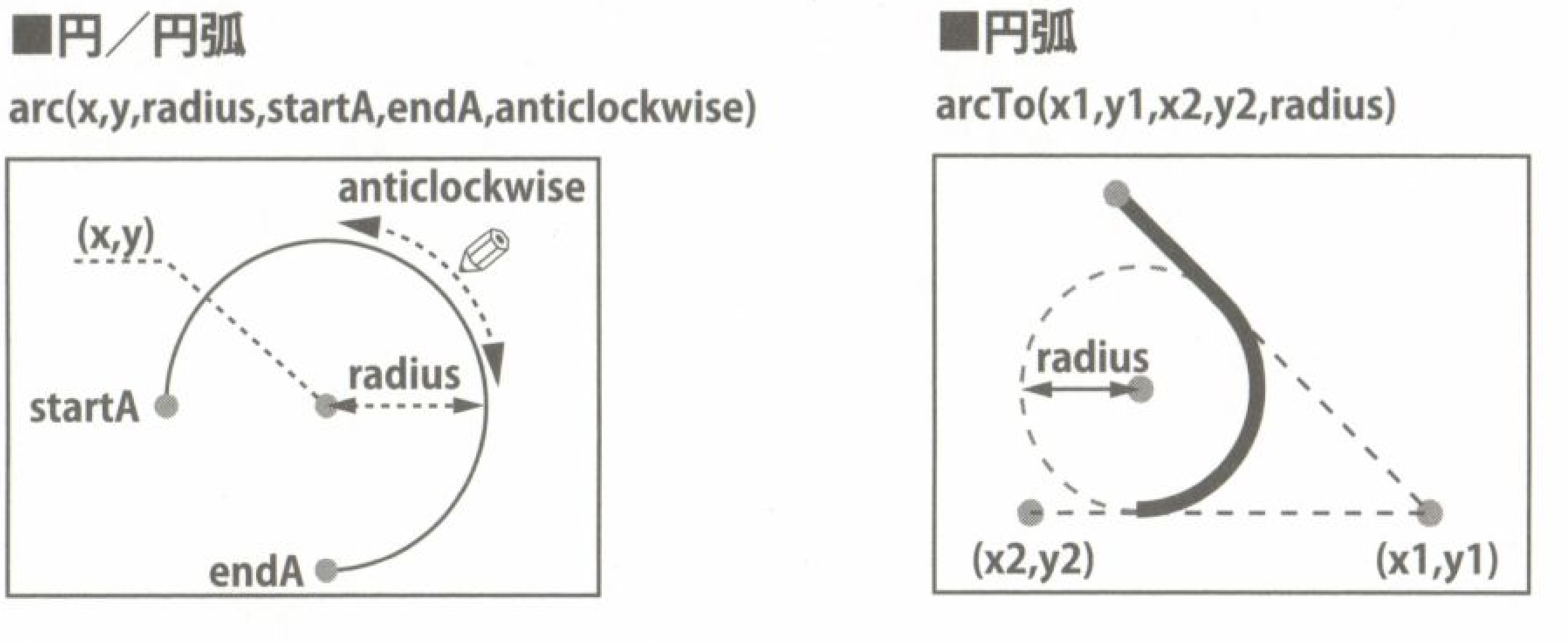
ONEPOINT 円や円弧の描画を行うには「arc()」メソッドや「arcTo()」メソッドを使う

円や円弧を描画するには、「arc()」メソッド、または、「arcTo()」メソッドを使います。「arc()」メソッドは、円の中心座標と半径、そして開始角度と終了角度を指定することができます。開始角度と終了角度を指定することで、円弧を描くことができます。また、開始角度、終了角度はラジアンで指定します。

「arcTo()」メソッドは、円弧と直線をまとめて描きます。

```
arc(x, y, radius, startAngle, endAngle, anticlockwise)
arcTo(x1, y1, x2, y2, radius)
```

なお、それぞれのパラメータは、次の図のようになります。

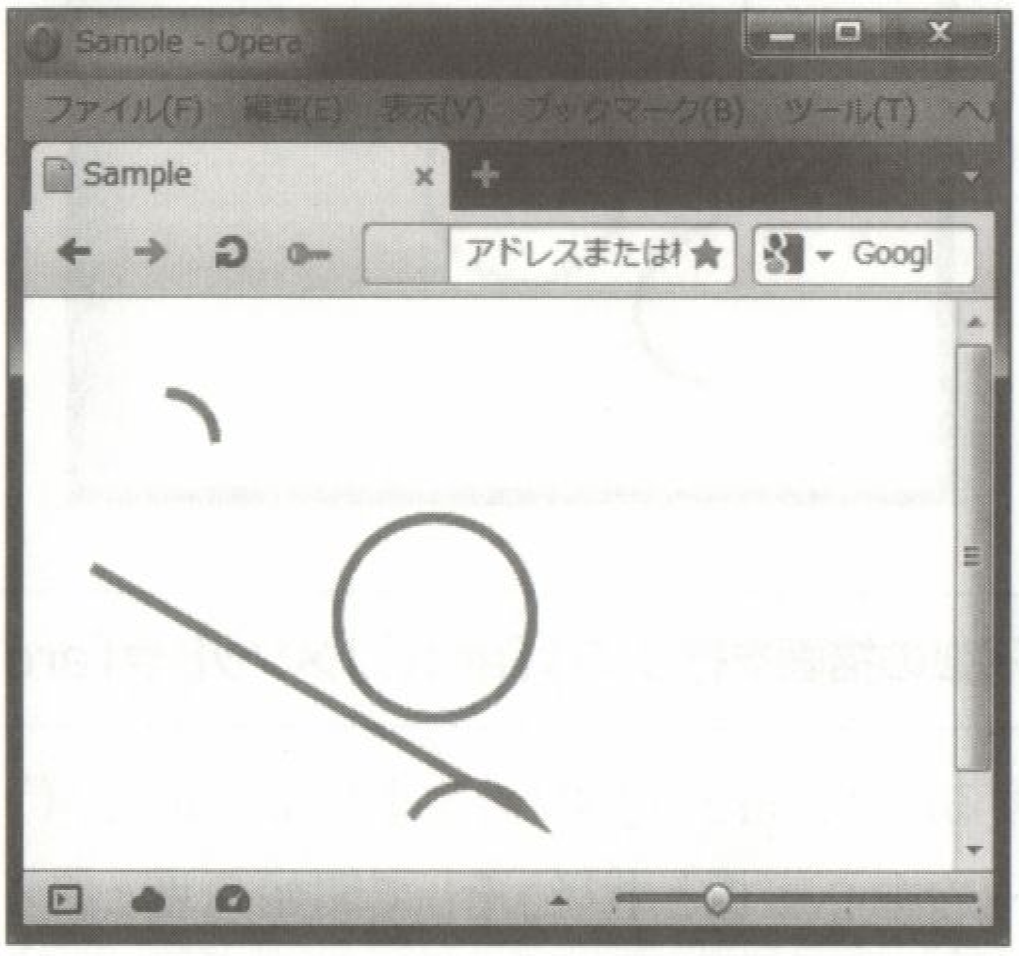


COLUMN Androidでの不具合

Android 2.0〜4.0の標準ブラウザでは、「arc()」メソッドでの時計回り/反時計回りの指定は無視されます。このため、ベクトルの向きによってくり抜く(ドーナツ型など)ことはできません。Android 4.1以降の標準ブラウザ(Chrome for Android)やFirefoxなど、他のブラウザなら正しく処理されます。

COLUMN Operaでの不具合

Operaでは「arcTo()」メソッドに不具合があり、パラメータによっては正しく描画されません。サンプルも正しく描画されず、Opera 12では、次の図のように描画されてしまいます。



SECTION-197

曲線を描く

ここでは、曲線を描く方法について解説します。

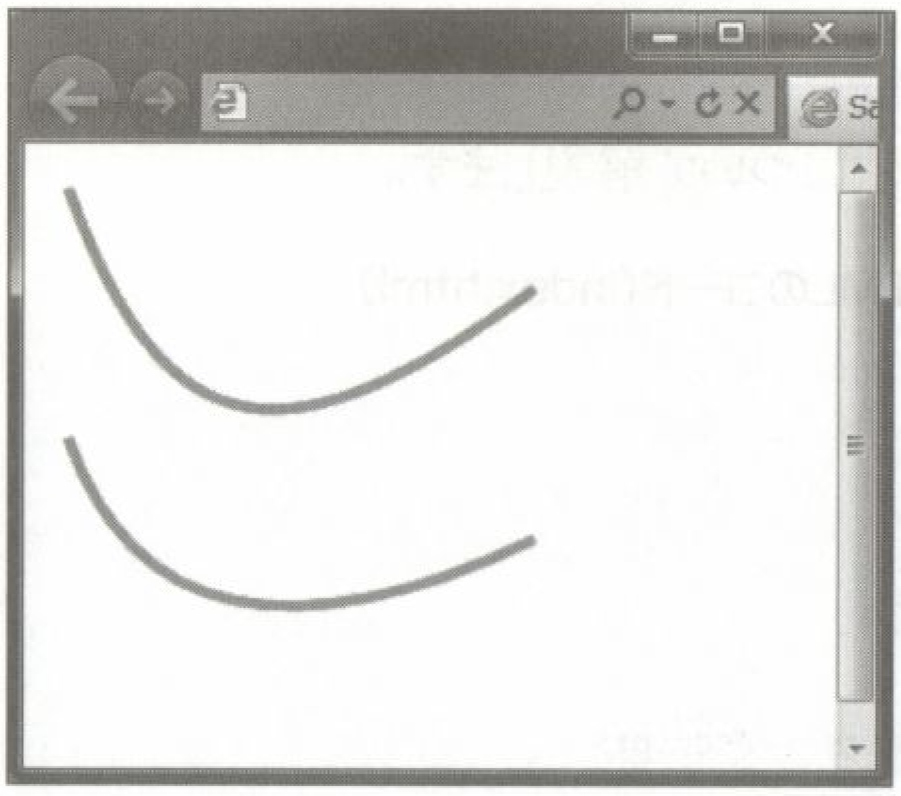
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色とサイズを設定
  context.strokeStyle = "red";
  context.lineWidth = 4;
  // 3次ベジエ曲線を描く
  context.beginPath();
  context.moveTo(10,10);
  context.bezierCurveTo(50,120, 100, 120, 200, 50);
  context.stroke();
  // 2次ベジエ曲線を描く
  context.beginPath();
  context.moveTo(10,110);
  context. quadraticCurveTo(50,220, 200, 150);
  context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、上に3次ベジエ曲線、下に2次ベジエ曲線が描画されます。



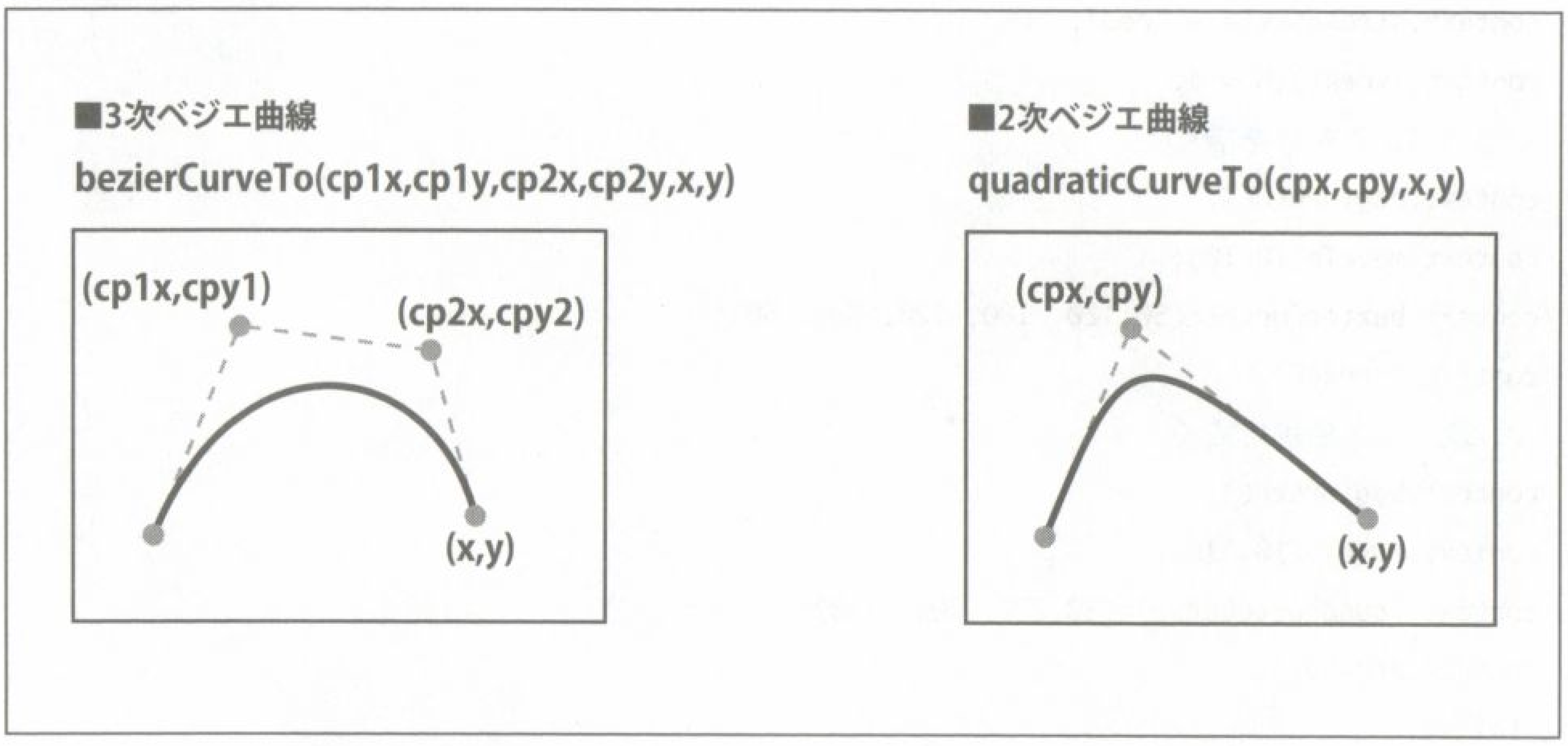
ONEPOINT 曲線の描画を行うには「bezierCurveTo()」メソッドや「quadraticCurveTo()」メソッドを使う

曲線を描画するには、「bezierCurveTo()」メソッド、または、「quadraticCurveTo()」メソッドを使います。「bezierCurveTo()」メソッドは、3次ベジエ曲線を描きます。3次ベジエ曲線には、開始点と終了点そして2つの制御点が必要です。このうち、開始点は、「moveTo()」メソッドを使って指定します。

「quadraticCurveTo()」メソッドは、2次ベジエ曲線を描きます。2次ベジエ曲線は、開始点と終了点と1つの制御点が必要です。開始点は、「moveTo()」メソッドで指定します。

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
quadraticCurveTo(cpx, cpy, x, y)
```

なお、それぞれのパラメータは、次の図のようになります。



SECTION-198

多角形を描く

ここでは、三角形や十二角形などの多角形を描く方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

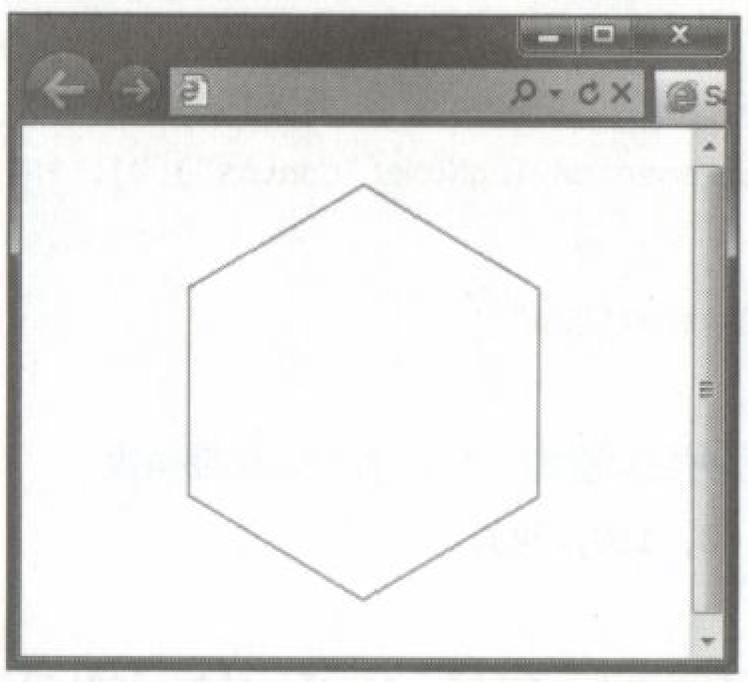
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 六角形を描く。n角形、半径、中心X座標、中心Y座標、初期角度
  drawPoly(6, 100, context, 160, 120, 30);
  // n角形を描く
  function drawPoly(n, size, context, baseX, baseY, startAngle){
    // Canvasの状態を保存
    context.save();
    // 指定された座標に移動
    context.translate(baseX, baseY);
    // 線の色とサイズを設定
    context.strokeStyle = "red";
    context.lineWidth = 1;
    // 新規にパスを作成
    context.beginPath();
    // 角度から初期座標を求める
    var degree = 360 / n;
    var rad = startAngle * Math.PI / 180;
    var x = size * Math.cos(rad);
    var y = size * Math.sin(rad);
```

```
    context.moveTo(x, y);
    // n回繰り返す
    for(var i=0; i<n; i++){
      // 角度を求める
      var angle = i * degree;
      rad = (angle + startAngle) * Math.PI / 180;
      // 角度からXY座標を求める
      x = size * Math.cos(rad);
      y = size * Math.sin(rad);
      // 直線の開始座標を指定
      context.lineTo(x, y);
    }
    // パスを閉じる
    context.closePath();
    // 線を描画
    context.stroke();
    // Canvasの状態を元に戻す
    context.restore();
  }
}, false);
```

ブラウザでindex.htmlを表示すると、赤い線で六角形が描画されます。



ONEPOINT　多角形を描画するには角度からX座標とY座標を求める

多角形を描画するには、角度からX座標とY座標を求め、直線でつなぎます。五角形であれば「360度÷5=72」で、角度を72度ずつ変化させ、X座標とY座標を求めます。角度からX座標とY座標を求めるには、次のようになります。

```
var rad = 角度 * Math.PI / 180;
var x = 半径 * Math.cos(rad);
var y = 半径 * Math.sin(rad);
```

これで得られたX座標とY座標を「lineTo()」メソッドを使ってパスを作成し、最後に「stroke()」メソッドで線を描画します。また、「fill()」メソッドであれば、塗りつぶされた多角形になります。

角丸四角形を描く

ここでは、角丸四角形を描く方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

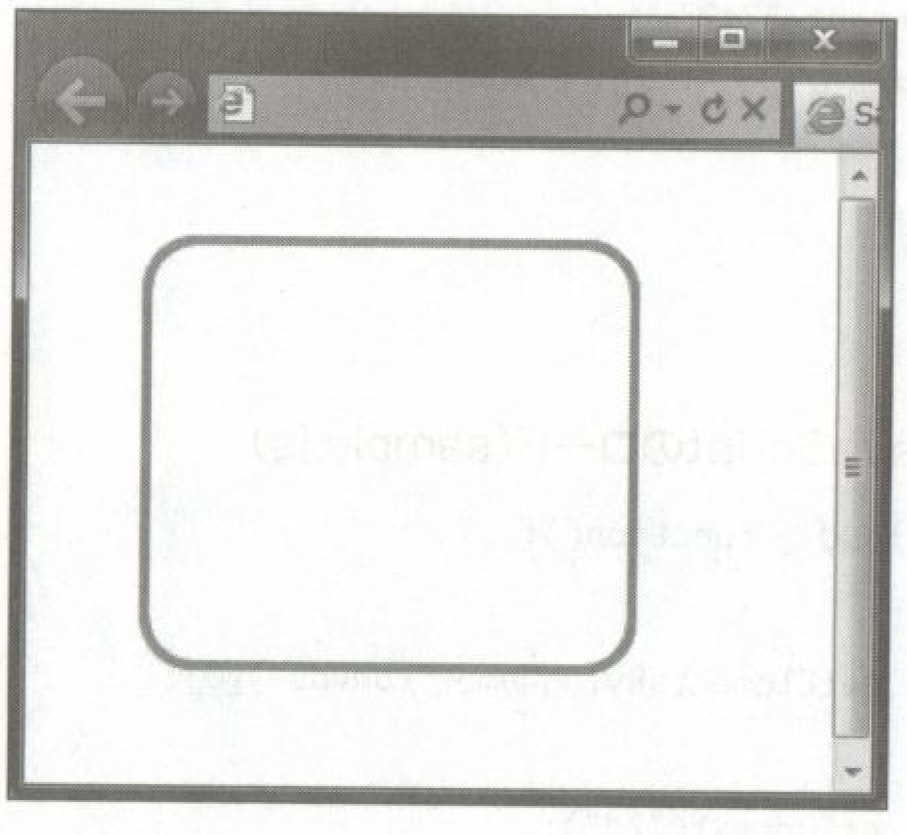
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色と幅を設定
  context.lineWidth = 4;
  context.strokeStyle = "red";
  // 角丸四角形を描く。x, y, 横幅, 縦幅, 角丸半径
  drawRoundRect(context, 40, 30, 200, 170, 20);
  // 角丸四角形を描くための関数
  function drawRoundRect(context, x, y, w, h, r){
    // 幅の小さい場合と大きい場合で半径を調整
    if (w<h){
      if ((w/2) < r){ r = w / 2; }
    }else{
      if ((h/2) < r){ r = h / 2; }
    }
    // 新規パスを作成
    context.beginPath();
    // 左上に移動
    context.moveTo(x+r, y);
    // 直線と角丸を描く
    context.lineTo(x+w-r, y);
```

```
    context.arc(x+w-r, y+r, r, -Math.PI/2, 0, false);
    context.lineTo(x+w, y+h-r);
    context.arc(x+w-r, y+h-r, r, 0, Math.PI/2, false);
    context.lineTo(x+r, y+h);
    context.arc(x+r, y+h-r, r, Math.PI/2, Math.PI, false);
    context.lineTo(x, y+r);
    context.arc(x+r, y+r, r, Math.PI, -Math.PI/2, false);
    // パスを閉じる
    context.closePath();
    // 線を描く
    context.stroke();
  }
}, false);
```

ブラウザでindex.htmlを表示すると、赤い線で角丸四角形が描画されます。



ONEPOINT

角丸四角形を描画するには「lineTo()」メソッドと「arc()」メソッドを組み合わせる

角丸四角形を描画するには、直線を描く「lineTo()」メソッドと角丸部分を描画する「arc()」メソッドを組み合わせます。canvasではパスを作成することができるので、これを利用します。「moveTo()」メソッド、「lineTo()」メソッド、「arc()」メソッドを順番に組み合わせて角丸四角形になるようにパスを作成します。作成したパスは「stroke()」メソッドで線だけの角丸四角形を描画、「fill()」メソッドで塗りつぶされた角丸四角形を描画することができます。

SECTION-200

ドーナツなどの中をくり抜いた図形を描く

ここでは、ドーナツなど中をくり抜いた状態の図形を描く方法について解説します。

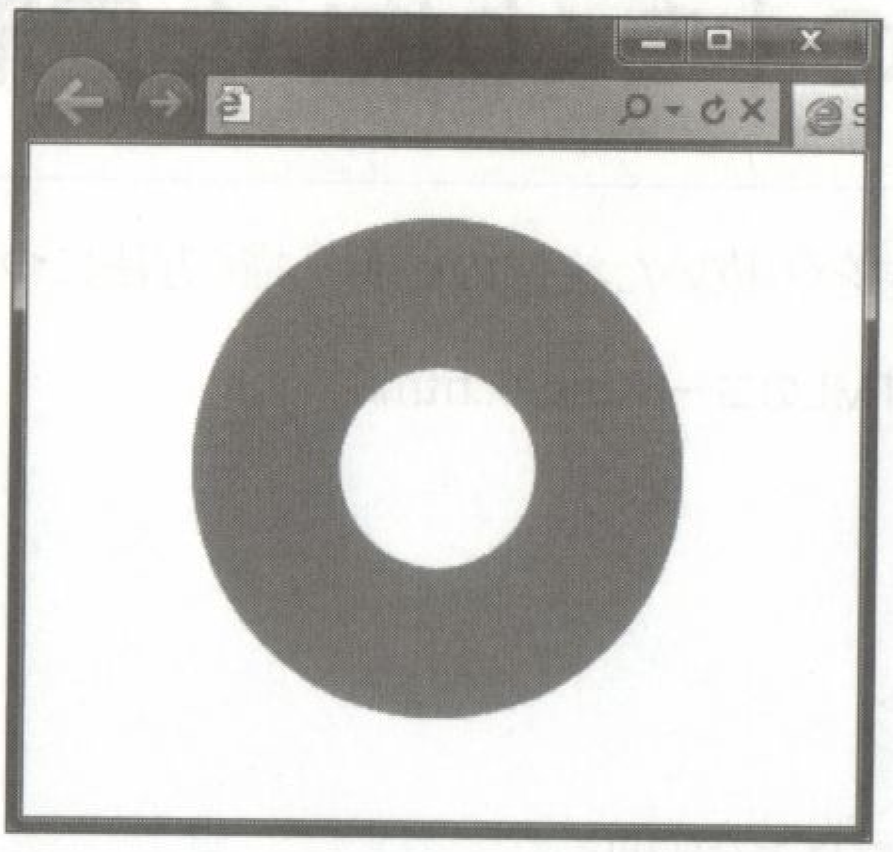
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色と幅を設定
  context.lineWidth = 4;
  context.fillStyle = "red";
  // ドーナツ型を描く。x, y, 外側の半径, 内側の半径
  drawDoughnut(context, 160, 120, 100, 40);
  // ドーナツ型を描くための関数
  function drawDoughnut(context, x, y, outR, inR){
    // 新規パスを作成
    context.beginPath();
    // 外側の円を描く
    context.arc(x, y, outR, 0, Math.PI*2, false);
    // 内側の円を描く(反時計回り)
    context.arc(x, y, inR, 0, Math.PI*2, true);
    // パスを閉じる
    context.closePath();
    // 線を描く
    context.fill();
  }
}, false);
```

ブラウザでindex.htmlを表示すると、赤く塗りつぶされたドーナツ型の図形が描画されます。



ONEPOINT 中をくり抜いた図形を描くにはパスのベクトルを考慮する

ドーナツ型のように中をくり抜いた図形を描くには、パスのベクトルを考慮します。Canvasでは作成したパスの向き(ベクトルの向き)が逆方向の場合、重なった部分がくり抜かれます。これを偶奇規則(ワインディングルール)といいます。

「arc()」メソッドの場合は、最後のパラメータによって描画方向(ベクトル)を指定することができます。これを利用することで、ベクトルの向きを変えて2つの円を重なるようにして半径だけ変えれば、ドーナツ型を描画することができます。

COLUMN Andoridでの不具合

Android 2.0～4.0の標準ブラウザでは、「arc()」メソッドでの時計回り/反時計回りの指定は無視されます。このため、ベクトルの向きによってくり抜く(ドーナツ型など)ことはできません。Android 4.1以降の標準ブラウザ(Chrome for Android)やFirefoxなど、他のブラウザなら正しく処理されます。

色と線のスタイルを設定する

ここでは、色と線のスタイルを設定する方法について解説します。

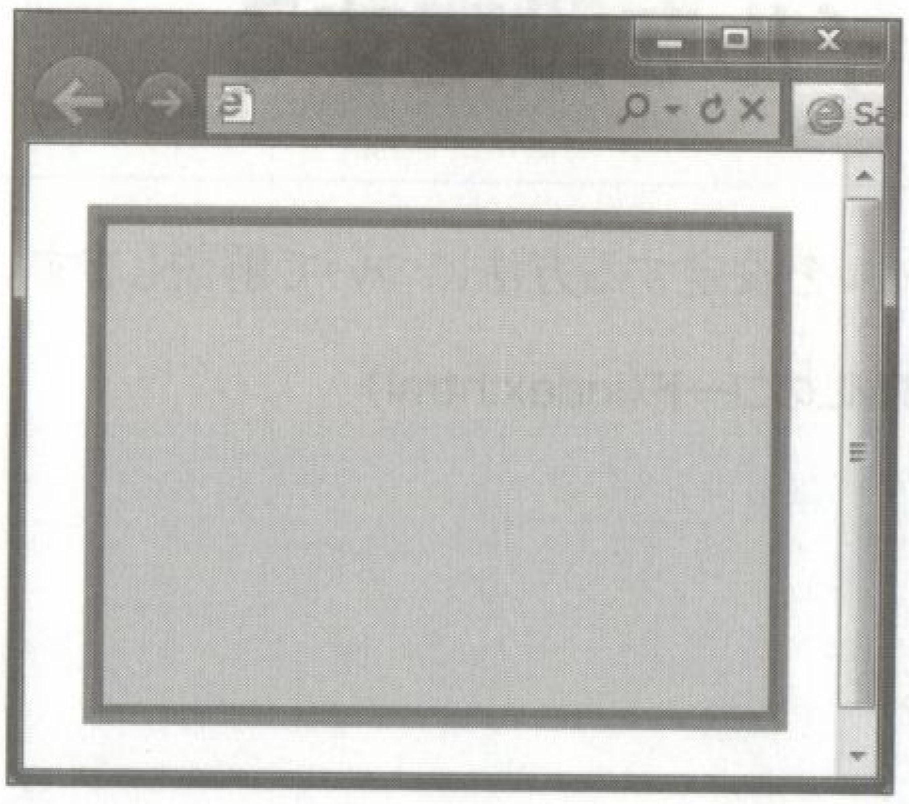
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線の色とサイズ、塗りの色を設定
  context.strokeStyle = "rgba(255,0,0, 0.75)";
  context.lineWidth = 8;
  context.fillStyle = "cyan";
  // パスで四角形を描く
  context.beginPath();
  context.rect(20,20, 280,200);
  context.closePath();
  context.fill();
  context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、色と線のスタイルが設定された四角形が描かれます。



ONEPOINT　スタイルの指定を行うには「strokeStyle」プロパティを使う

パスで構築された図形の線のスタイルを設定するには、「strokeStyle」プロパティにカラー名やカラーコードを指定します。色は、CSS3で指定できる形式を使用できます(887ページ参照)。

線でなく、塗りつぶしスタイルを設定するには、「fillStyle」プロパティにカラー名やカラーコードを指定します。「fillStyle」プロパティも、「strokeStyle」プロパティと同様に、CSS3で指定できる形式を使用できます。

SECTION-202

不透明度を設定する

ここでは、不透明度を設定する方法について解説します。

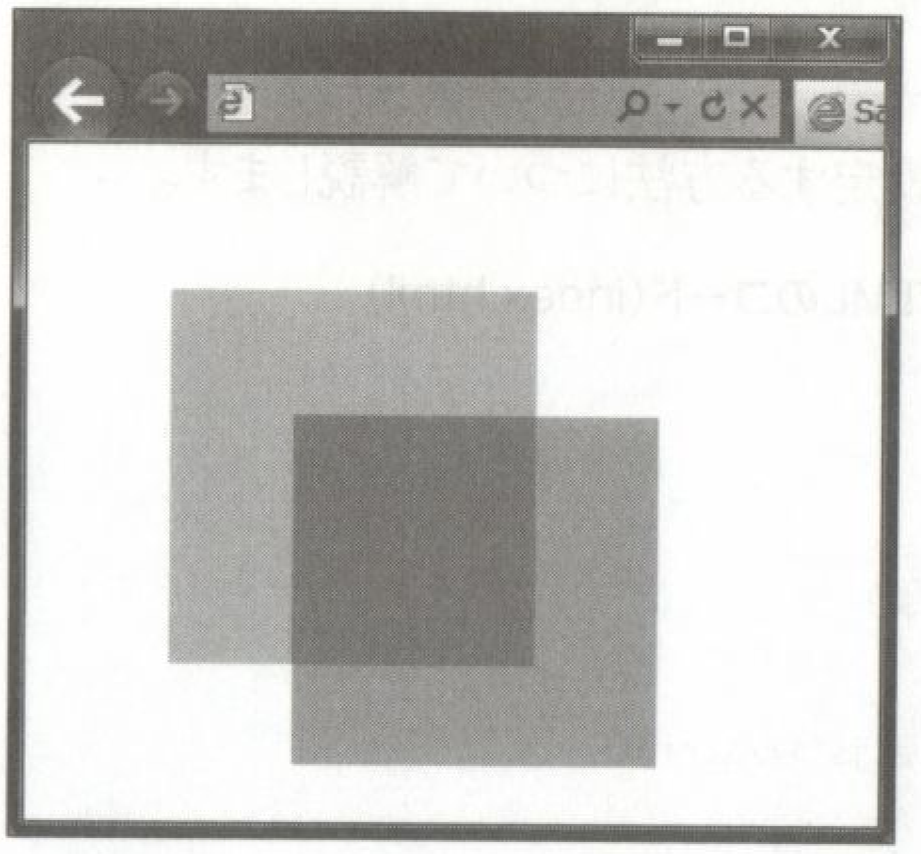
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 全体の不透明度を設定
  context.globalAlpha = 0.25;
  // 塗りの色を設定
  context.fillStyle = "red";
  // 四角形を描画
  context.fillRect(50, 50, 150, 150);
  // 塗りの色を設定
  context.fillStyle = "blue";
  // 四角形を描画
  context.fillRect(100, 100, 150, 150);
}, false);
```

ブラウザでindex.htmlを表示すると、半透明になった赤色の四角形と青色の四角形が表示されます。重なった部分は、赤と青の半透明で描画されています。



ONEPOINT　不透明度の指定を行うには「globalAlpha」プロパティを使う

不透明度を指定するには、コンテキストの「globalAlpha」プロパティに「0.0」～「1.0」までの値を設定します。「0.0」が完全な透明、「1.0」が完全な不透明になります。「globalAlpha」を設定すると、以後に描画される図形は、すべて影響を受けます。「globalAlpha」プロパティで設定した値を元に戻す場合は、「save()」メソッドと「restore()」メソッドを利用します。「save()」メソッドは、直前の不透明度も保存しておくことができるためです。

また、全体の不透明度ではなく、一部だけに適用したい場合は、「fillStyle」プロパティか「strokeStyle」プロパティに、「rgba()」形式で色と不透明度を一括で指定します。

SECTION-203

描画モードを設定する

ここでは、描画モードを設定する方法について解説します。

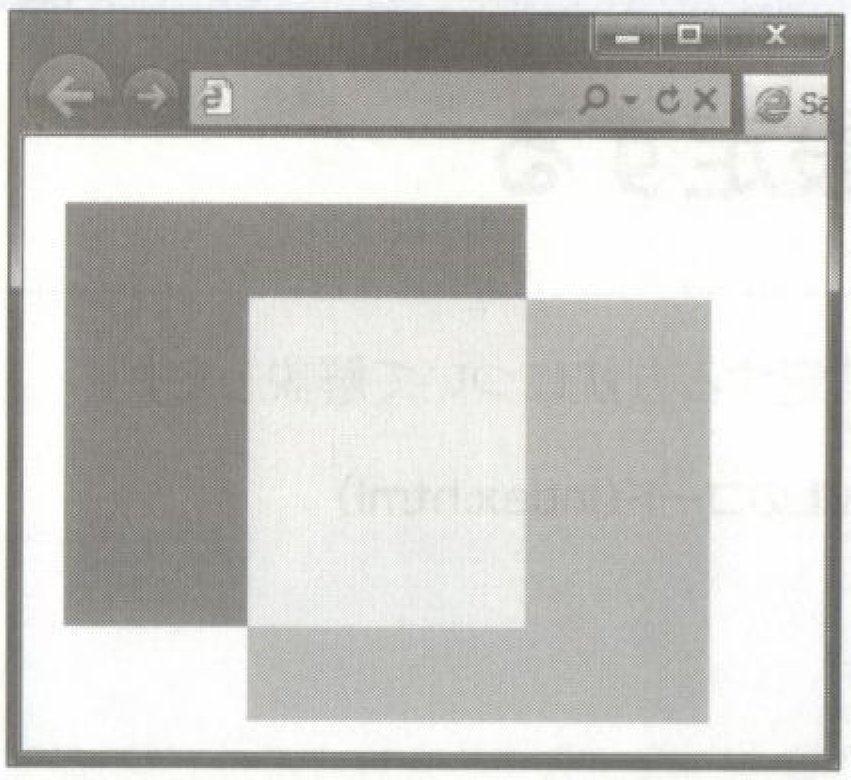
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 赤色に設定
  context.fillStyle = "red";
  // デフォルトのモード(source-over)で四角形を描画
  context.fillRect(10,20, 200, 180);
  // 緑色に設定
  context.fillStyle = "#00ff00";
  // 描画モードをlighterに設定
  context.globalCompositeOperation = "lighter";
  // 四角形を描画
  context.fillRect(90, 60, 200, 180);
}, false);
```

ブラウザでindex.htmlを表示すると、lighterの描画モードで赤い四角形と緑色の四角形を描画されます。重なった部分は、合成された明るい色(黄色)で描画されています。



ONEPOINT 描画モードの指定を行うには「globalCompositeOperation」プロパティを使う

描画モードを指定するには、コンテキストの「globalCompositeOperation」プロパティに、次の表に示す文字列を指定します。描画モードを設定した際、ブラウザによって結果が異なることがあります。

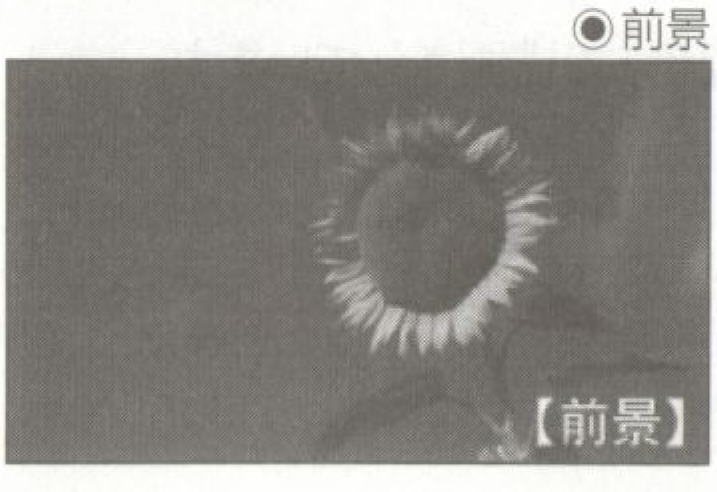
文字列	説明
source-atop	背景はそのままで背景と前景が重なった部分だけに前景を描画
source-in	背景と前景が重なった部分だけに前景を描画
source-out	背景が重なっていない部分だけに前景を描画
source-over（デフォルト）	前景をそのまま描画
destination-atop	背景と前景が重なった場合、重なった部分に背景を描画
destination-in	背景と前景が重なった部分だけに背景を描画
destination-out	前景が重なっていない部分だけに背景を描画
destination-over	背景と重なった部分は前景を描画しない
lighter	重なった部分をより明るく描画
copy	前景をコピー
xor	背景と前景が重なった部分は描画されない
vendorName-operationName	ベンダー独自の描画モード

それぞれの文字列を指定した場合の結果は、次の図のようになります。

◉背景



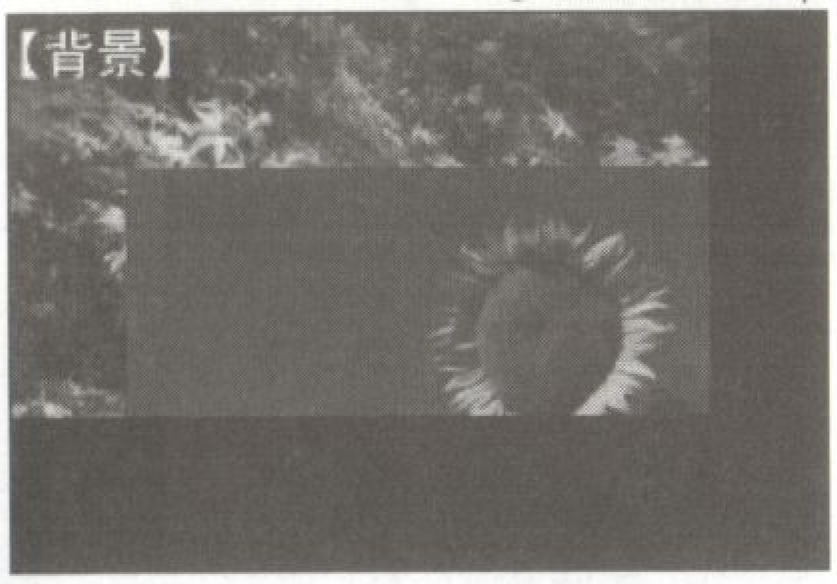
◉前景



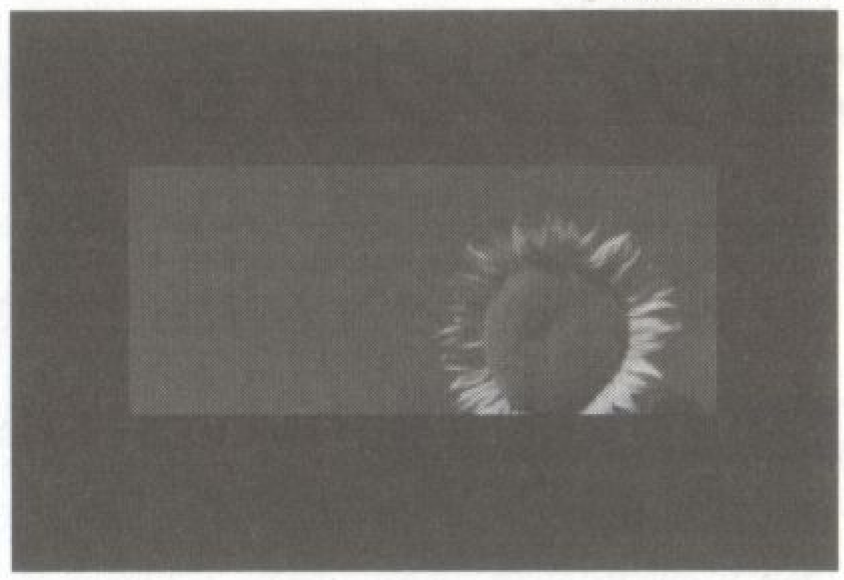
◉source-over（デフォルト）



◉source-atop



◉source-in



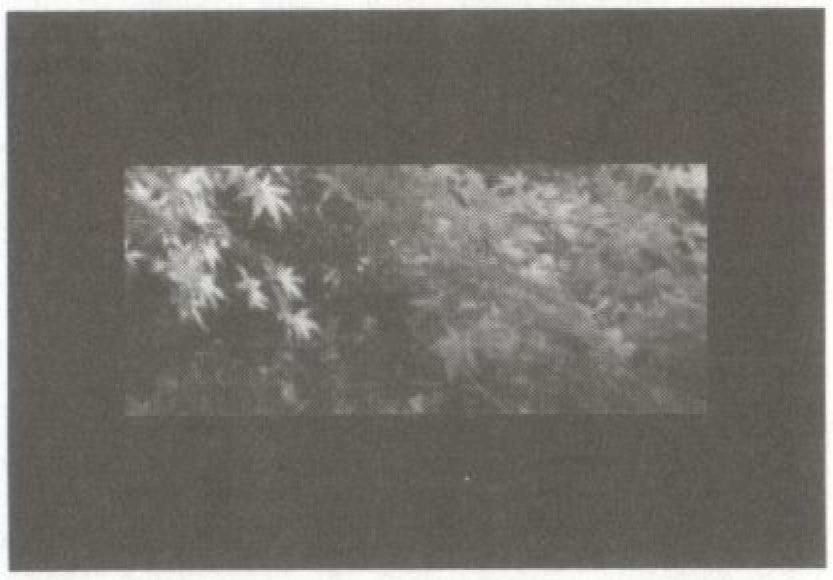
◉source-out



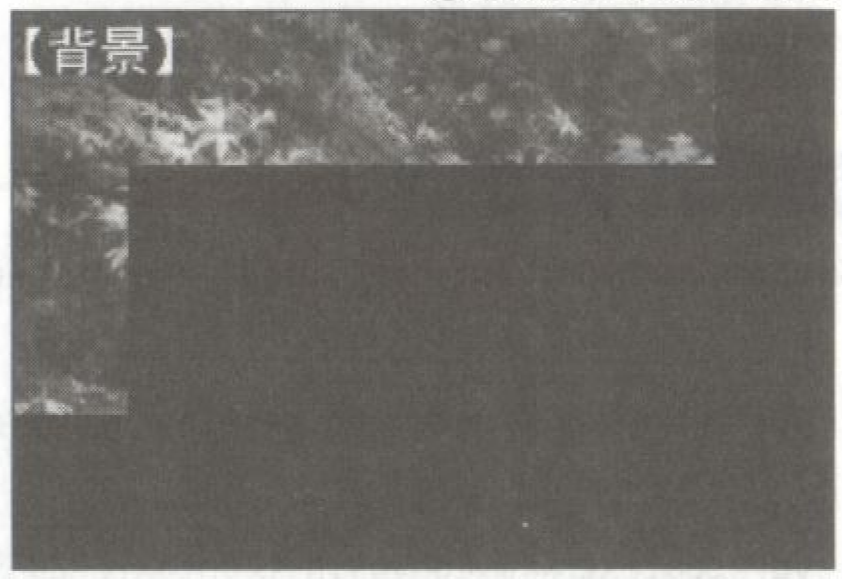
◉destination-atop



◉destination-in



◉destination-out



◉destination-over



◉lighter



◉copy



◉xor



SECTION-204

クリッピングする

ここでは、クリッピングする方法について解説します。

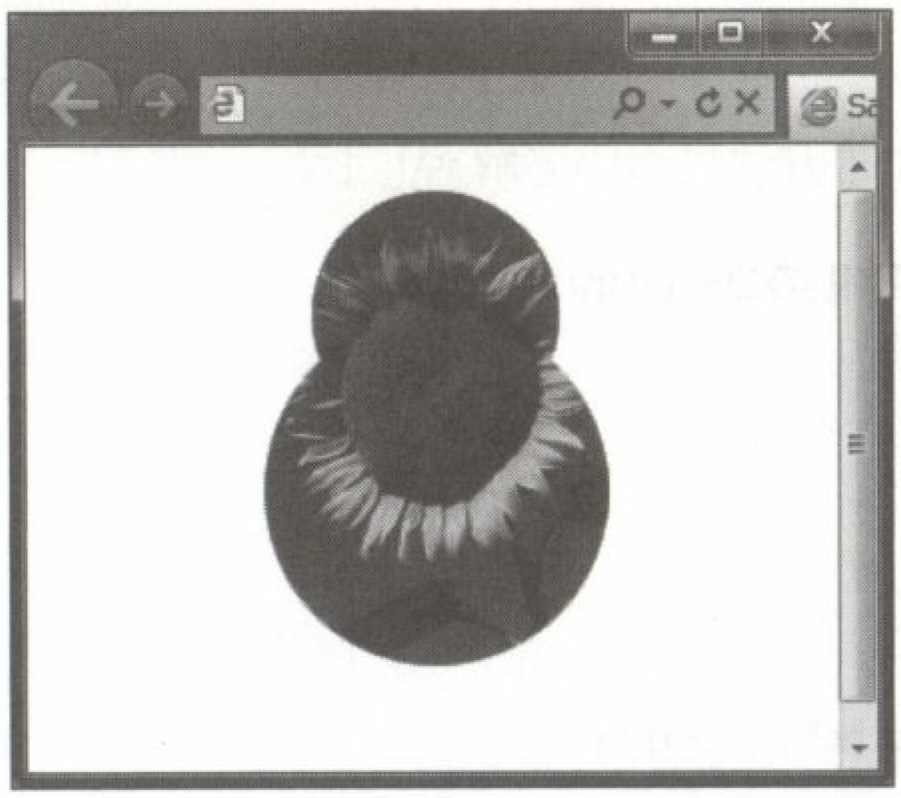
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // パスで正円を2つ作成してクリッピング領域にする
  context.beginPath();
  context.arc(160,60, 50, 0, Math.PI*2, false);
  context.arc(160,130, 70, 0, Math.PI*2, false);
  context.clip();
  // 画像を描画する
  var imgObj = new Image();
  imgObj.src = "images/sunflower.jpg";
  imgObj.onload = function(){
    context.drawImage(this, -70, 0);
  }
}, false);
```

ブラウザでindex.htmlを表示すると、2つの円の中に画像が描画されます。2つの円からはみ出した部分は描画されません。



ONEPOINT　クリッピング領域の指定を行うには「clip()」メソッドを使う

クリッピング領域を設定するには、パスを構築するメソッド(「moveTo()」「lineTo()」「arc()」「arcTo()」「rect()」「bezierCurveTo()」「quadraticCurveTo()」メソッド)を使います。パスでクリッピングする領域を設定した後に、「clip()」メソッドを実行します。これによって、パス部分が描画範囲に設定されます。範囲からはみ出した部分は図形でも画像でも描画されません。

また、設定したクリッピング範囲を解除するには、クリッピング領域を設定する前に、「save()」メソッドでCanvasの情報を保存しておきます。その後、「restore()」メソッドを使えば、設定したクリッピング領域を解除することができます。

なお、2012年のCanvasの最新仕様では、「clip()」メソッドで設定したクリッピング範囲を解除する「resetClip()」メソッドが検討されています。

SECTION-205

文字を描画する

ここでは、文字を描画する方法について解説します。

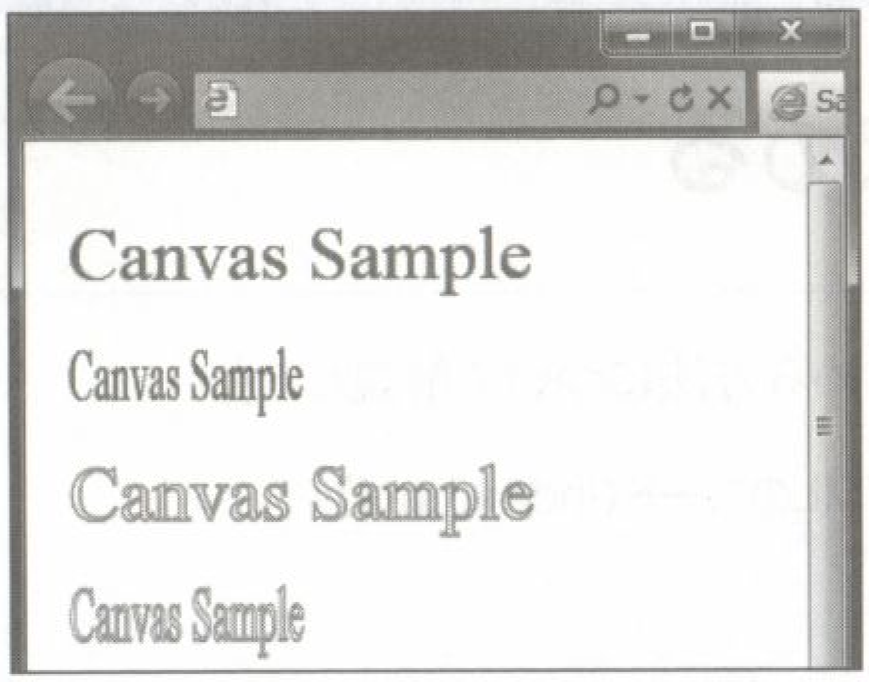
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // フォントとサイズを設定する
  context.font = "24pt 'Tahoma-Bold'";
  // 文字の色を設定する
  context.fillStyle = "blue";
  // アウトライン文字の線の色を設定する
  context.strokeStyle = "red";
  // 文字を描画する
  context.fillText("Canvas Sample", 10, 50);
  // 横幅を指定して文字を描画する(Safari 5は非対応)
  context.fillText("Canvas Sample", 10, 100, 100);
  // アウトライン文字を描画する
  context.strokeText("Canvas Sample", 10, 150);
  // 横幅を指定してアウトライン文字を描画する
  context.strokeText("Canvas Sample", 10, 200, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、Canvasに指定したサイズと色で文字が描画されます。



ONEPOINT
文字を描画するには「fillText()」メソッドや「strokeText()」メソッドを使う

文字を描画するには、「fillText()」メソッド、または、「strokeText()」メソッドを使います。「fillText()」は塗りつぶされた文字を、「stroketext()」は線だけの文字(アウトライン文字/袋文字)を描画します。「fillText()」メソッドと「strokeText()」メソッドのパラメータは同じです。最初のパラメータが、描画する文字列になります。2番目がX座標、3番目がY座標になりますが、Y座標は文字のベースラインが基準になります。

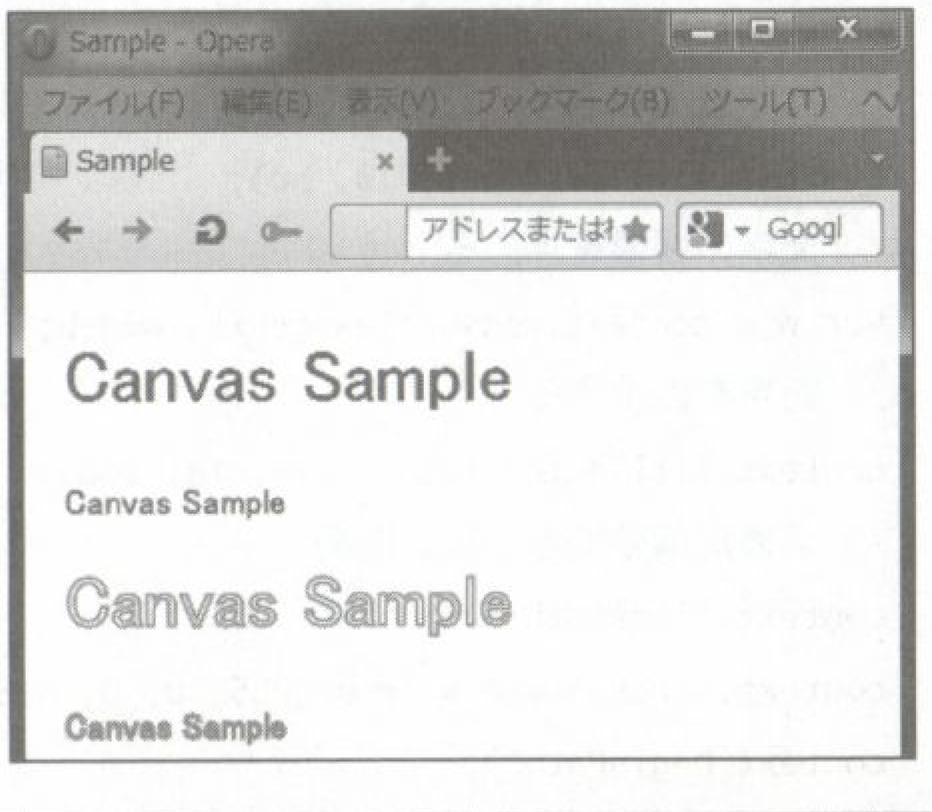
```
fillText(text, x, y, [width])
strokeText(text, x, y, [width])
```

4番目のパラメータは、文字を描画する範囲(横幅)を指定します(単位はピクセル)。横幅を指定すると、その範囲内に文字が収まるように調整されます。これによって、文字が縦長(長体)になります。

なお、Canvas仕様ver 5では、パス上にテキストを描画する機能(「addText()」メソッド)も追加されています。

COLUMN
Safari 5とOperaでの不具合

Safari 5とOpera 11.0～12.0で文字の描画範囲を示す横幅を指定した場合、正しく動作しません。Safari 5では指定した幅は無視され文字が描画されます。Opera 11.0～12.0では文字が長体にならず、範囲に収まるように正体のまま文字全体が小さくなります。なお、Safari 6ではこの不具合は解消されています。



関連項目 ▶▶▶

● 文字の幅を求める ……… p.472
● 文字の位置揃えを指定する ……… p.474

SECTION-206

文字の幅を求める

ここでは、文字の幅を求める方法について解説します。

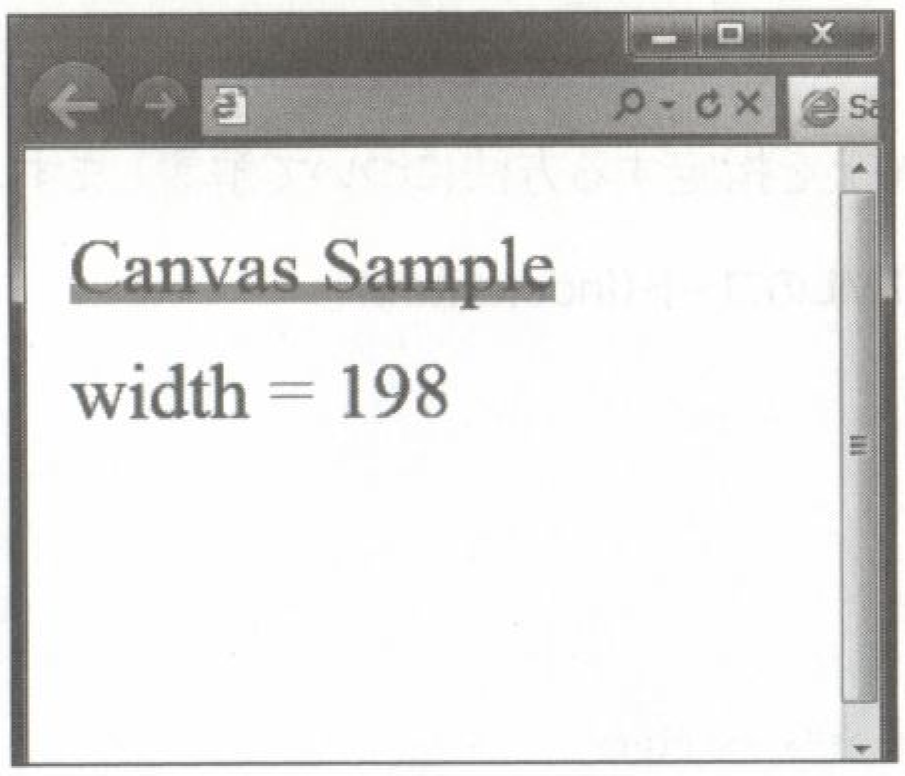
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // フォントとサイズおよび色を設定する
  context.font = "24pt 'Tahoma-Bold'";
  context.fillStyle = "blue";
  // 描画する(文字幅を計算する)文字
  var text = "Canvas Sample";
  // 文字を描画する
  context.fillText(text, 10, 50);
  // 文字の横幅を求める
  var w = context.measureText(text).width;
  // 結果を表示する
  context.fillText("width = "+w, 10, 100);
  // 求めた横幅で赤い線を描画
  context.lineWidth = 8;
  context.strokeStyle = "rgba(255, 0, 0, 0.5)";
  context.beginPath();
  context.moveTo(10,50);
  context.lineTo(10+w, 50);
  context.stroke();
}, false);
```

ブラウザでindex.htmlを表示すると、文字が描画され、その下に文字の幅が表示されます。その文字幅に従って、文字の上に赤い線が描画されます。



ONEPOINT 文字幅を求めるにはテキストメトリクスオブジェクトの「width」プロパティを使う

描画する文字幅を求めるには、あらかじめ、コンテキストの「font」プロパティに描画する文字のサイズと書体(フォント)を設定しておきます。次に、「mesureText()」メソッドを使ってテキストメトリクスオブジェクトを取得します。このテキストメトリクスオブジェクトの「width」プロパティに、文字幅がピクセル数で返されます。ただし、書体(フォント)によっては、求めた描画幅よりも文字がはみ出す場合があります。

また、テキストメトリクスオブジェクトには「width」プロパティしかなく、文字の縦幅など他のサイズを求めることはできません。

COLUMN 検討中のテキストメトリクスのプロパティ

テキストの情報が横幅しか取得できないのは、実用度に欠けます。このため、次のようなプロパティが検討されています。

- actualBoundingBoxLeft
- actualBoundingBoxRight
- fontBoundingBoxAscent
- fontBoundingBoxDescent
- emHeightAscent
- emHeightDescent
- hangingBaseline
- alphabeticBaseline
- ideographicBaseline

関連項目 ▶▶▶

● 文字を描画する ………… p.470
● 文字の位置揃えを指定する ………… p.474

SECTION-207

文字の位置揃えを指定する

ここでは、文字の位置揃えを指定する方法について解説します。

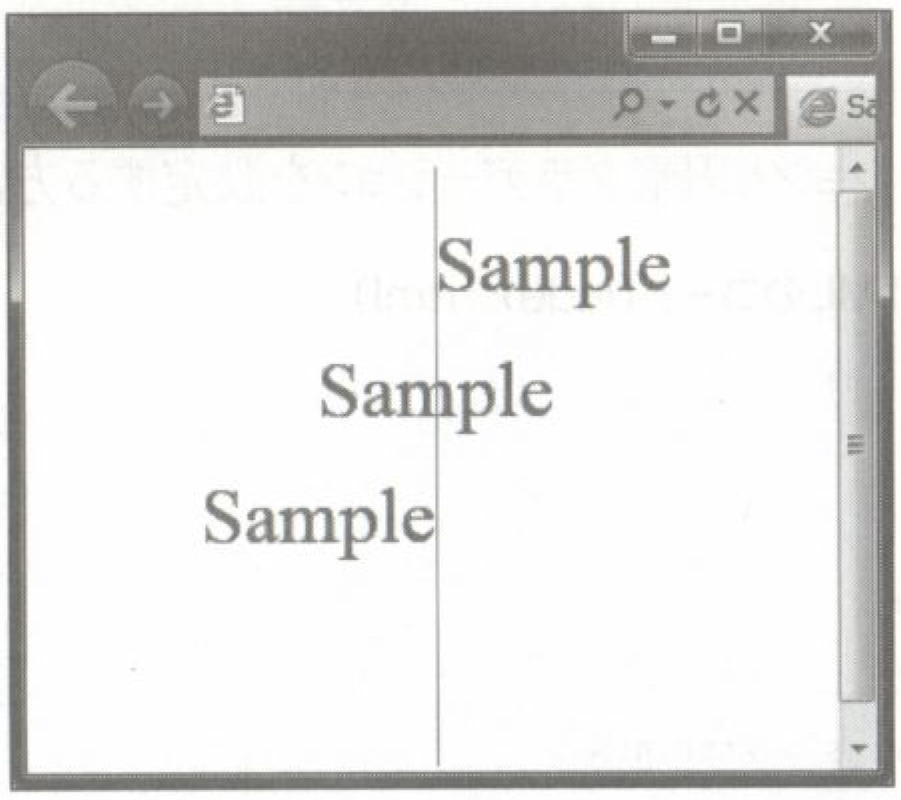
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // フォントとサイズおよび色を設定する
  context.font = "24pt 'Tahoma-Bold'";
  context.fillStyle = "blue";
  // 描画する文字
  var text = "Sample";
  // 文字を左揃えで描画する
  context.textAlign = "left";
  context.fillText(text, 160, 50);
  // 文字を中揃えで描画する
  context.textAlign = "center";
  context.fillText(text, 160, 100);
  // 文字を右揃えで描画する
  context.textAlign = "right";
  context.fillText(text, 160, 150);
  // 中心に線(四角形)を描画
  context.fillStyle = "red";
  context.fillRect(160, 0, 1, 240);
}, false);
```

ブラウザでindex.htmlを表示すると、上から左揃え、中揃え、右揃えで文字が描画され、基準となるX座標に赤い線が描画されます。



ONEPOINT **文字の位置揃えを指定するには「textAlign」プロパティを使う**

文字の位置揃えを指定するには、コンテキストの「textAlign」プロパティに、次の表に示す文字列を設定します。「start」「end」は右から左に描画するアラビア文字などの場合には、「left」「right」とは異なった結果になります。「start」を指定すると、開始文字の右側が基準となり、左側に文字が描画されます。「end」は「start」の逆で、最後の文字の左側に文字が揃えられます。

文字列	説明
left	左揃え
center	中揃え(センタリング)
right	右揃え
start	筆記方向の開始点で揃える
end	筆記方向の終了点で揃える

関連項目 ▶▶▶

●文字を描画する …… p.470
●文字の幅を求める …… p.472

SECTION-208

グラデーションを設定する

ここでは、線形グラデーションと円形グラデーションを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

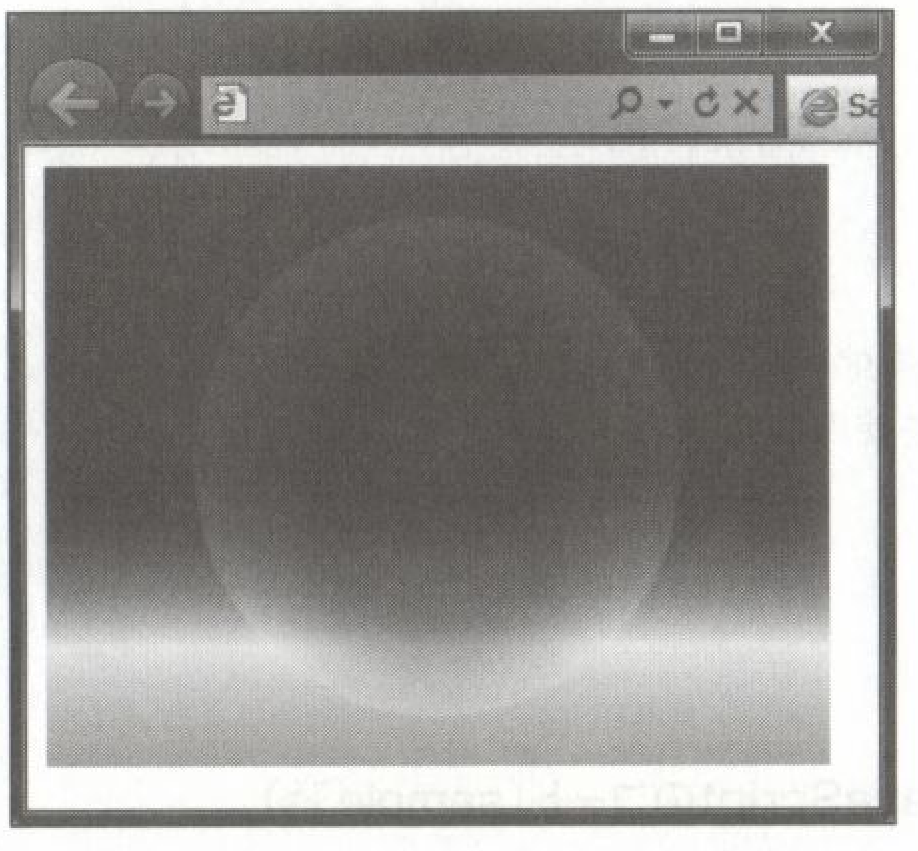
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 線形グラデーションを設定する
  var grad1 = context.createLinearGradient(0,0, 0, 240);
  // 開始点の色を指定する
  grad1.addColorStop(0, "rgb(0,50,255)");
  // 中間点1の色を指定する
  grad1.addColorStop(0.5, "rgb(0,0,255)");
  // 中間点2の色を指定する
  grad1.addColorStop(0.8, "yellow");
  // 終了点の色を指定する
  grad1.addColorStop(1, "orange");
  // 四角形をグラデーションで描画
  context.fillStyle = grad1;
  context.fillRect(0, 0, 320, 240);
  // 円形グラデーションを設定する
  var grad2 = context.createRadialGradient(160, 120, 20, 160, 120, 100);
  // 開始点の色を指定する
  grad2.addColorStop(0, "rgba(255,0,0, 0)");
  // 中間点1の色を指定する
  grad2.addColorStop(0.5, "rgba(255,0,0, 0.5)");
```

```
    // 終了点の色を指定する
    grad2.addColorStop(1, "rgba(255,255,255, 0.25)");
    // 正円をグラデーションで描画
    context.fillStyle = grad2;
    context.arc(160, 120, 100, 0, Math.PI*2, false);
    context.fill();
}, false);
```

ブラウザでindex.htmlを表示すると、Canvas全体に線形グラデーション、中心に円形グラデーションが描画されます。



ONEPOINT　グラデーションを描画するには「createLinearGradient()」メソッドや「createRadialGradient()」メソッドを使う

Canvasでは、線形グラデーションと円形グラデーションを描画するメソッドがあります。線形グラデーションは、「createLinearGradient()」メソッドを使います。パラメータには、グラデーションの開始座標と終了座標を指定します。生成されたグラデーションオブジェクトに色を設定するには、「addColorStop()」メソッドを使います。「addColorStop()」メソッドの最初のパラメータに、色の位置を指定します。「0」であれば0%の位置、「0.5」なら50%の位置、「1.0」なら100%の位置を示します。グラデーションオブジェクトを「fillStyle」プロパティに設定した後で「fill()」メソッドを使うと、図形がグラデーションで塗りつぶされます。

円形グラデーションは、「createRadialGradient()」メソッドを使います。パラメータには、開始座標と円の半径、終了座標と終了円の半径を指定します。色の指定は、線形グラデーションと同様に、「addColorStop()」メソッドを使います。カラーは、サンプルのように、不透明度を利用したグラデーションもできます。

画像を描画する

ここでは、画像を描画する方法について解説します。

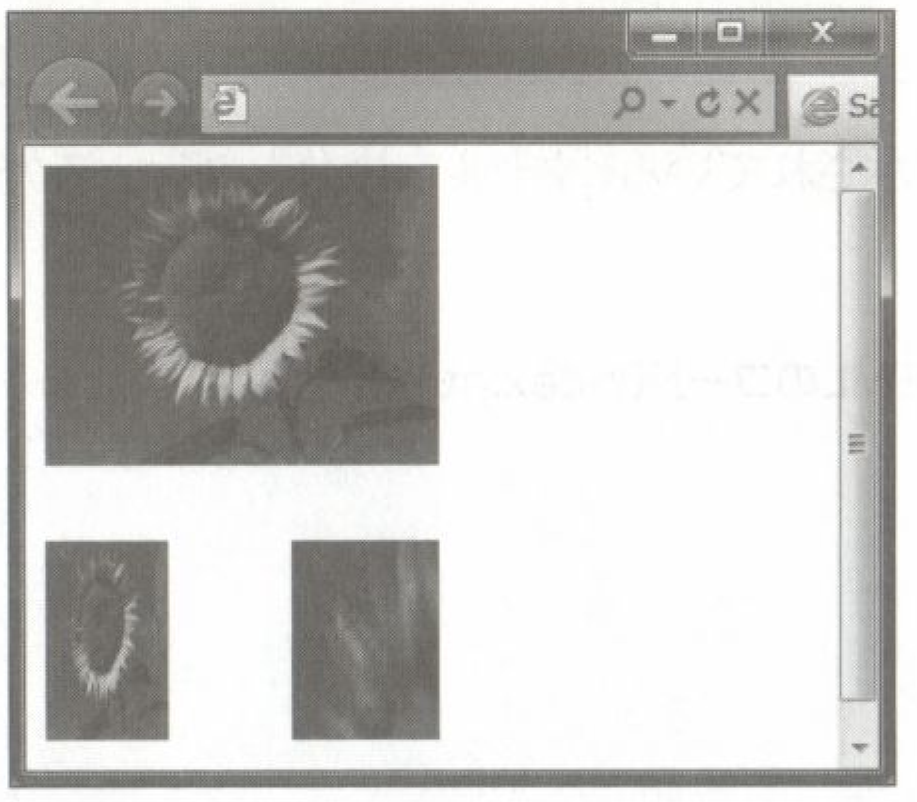
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 画像を描画する
  var imgObj = new Image();
  imgObj.src = "images/sunflower.jpg";
  imgObj.onload = function(){
    // 座標を指定して元のサイズで描画
    context.drawImage(this, 0, 0);
    // 座標とサイズを指定して描画
    context.drawImage(this, 0, 150, 50, 80);
    // 元画像の一部をクリッピングした後で指定した座標とサイズで描画
    context.drawImage(this, 30, 10, 40, 20, 100, 150, 60, 80);
  }
}, false);
```

ブラウザでindex.htmlを表示すると、3つの画像が描画されます。それぞれ元のサイズ、位置とサイズを指定、画像の一部をピックアップして描画しています。



ONEPOINT **画像を描画するには「drawImage()」メソッドを使う**

Canvasに画像を描画するには、「drawImage()」メソッドを使います。「drawImage()」メソッドは、「位置を指定して元のサイズで描画」「位置とサイズを指定して描画」「元画像の一部をクリッピング（切り抜き）した後、指定した位置とサイズで描画」の3種類で画像を描画することができます。それぞれ指定するパラメータは、次のようになります。

◉座標を指定して元のサイズで描画

```
drawImage(image, x, y)
```

◉座標とサイズを指定して描画

```
drawImage(image, x, y, width, height)
```

◉元画像の一部をクリッピングした後で指定した座標とサイズで描画

```
drawImage(image, sourceX, sourceY, sourceWidth, sourceHeight, destinationX, destinationY,
 destinationWidth, destinationY)
```

なお、描画する画像は、あらかじめ読み込まれていないといけません。読み込まれていない場合は、何も描画されません。また、DOM上にある「img」要素を指定することもできます。映像を描画する場合は「video」要素、CanvasにCanvasを描画する場合は「canvas」要素を指定します。

ただし、Android 1.6〜4.0/iOS 5で「drawImage()」の最初のパラメータに「video」要素を指定しても、期待通りに映像は描画されません。

SECTION-210

グレースケール画像にする

ここでは、Canvasに表示されているピクセルを操作しグレースケール画像に方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="グレースケールに加工" id="effect">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0, myCanvas.width, myCanvas.height);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // グレースケールに変換する関数を呼び出す
    toGray(context, myCanvas.width, myCanvas.height);
  }, false);
  // グレースケールに加工する関数
  function toGray(context, w, h){
    // Canvas内のピクセルデータを読み出す
    var pixelData = context.getImageData(0,0, w, h).data;
```

```
    // 縦方向のピクセル数だけ繰り返す
    for(var y=0; y<h; y++){
      // 横方向のピクセル数だけ繰り返す
      for(var x=0; x<w; x++){
        // 処理するピクセルの配列内での位置を計算する
        // 配列要素は「赤」「緑」「青」「不透明度(α)」の順番
        var pointer = (y * w + x) * 4;
        // 色データを読み出し
        var red = pixelData[pointer+0];
        var green = pixelData[pointer+1];
        var blue = pixelData[pointer+2];
        var alpha = pixelData[pointer+3];
        // 緑×0.6+赤×0.3+青×0.1で計算(おおまかに計算)
        var gray = Math.floor(green * 0.6 + red * 0.3 + blue * 0.1);
        // 計算結果を配列に入れる。同じ輝度にするとグレーになる
        pixelData[pointer+0] = gray;
        pixelData[pointer+1] = gray;
        pixelData[pointer+2] = gray;
        pixelData[pointer+3] = alpha;
      }
    }
    // 処理結果をCanvasに描画するために新規にピクセルデータを生成
    var output = context.createImageData(w, h);
    // ピクセルデータの総数だけ繰り返す
    for(var i=0; i<w*h*4; i++){
      // 操作したピクセルを入れる
      output.data[i] = pixelData[i];
    }
    // Canvasに描画する
    context.putImageData(output, 0,0);
  }
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、Canvasに描画されている画像がグレースケール画像に加工されて表示されます。





ONEPOINT

グレースケール画像にするには ピクセルデータを読み出して計算式に従って変換する

Canvasに描画されている画像を加工してグレースケール画像にするには、ピクセルデータを読み出し、計算式に従って変換します。ピクセルデータは、「getImageData()」メソッドを使って呼び出したオブジェクトの「data」プロパティに配列として格納されています。この配列には、色の輝度が「赤」「緑」「青」「不透明度(α)」の順番で格納されています。

グレースケールにするには、「緑×0.6+赤×0.3+青×0.1」で計算します(グレースケールへの変換式は他にもあるが、ここでは最も省略された計算式を使っている)。計算した結果が輝度になるので、赤青緑の輝度として再度、配列に代入します。

処理が終わった画像は、そのままではCanvas上には反映されません。そこで「createImageData()」メソッドを使ってCanvasと同じサイズのピクセルデータ(配列)を作成し、そこに処理済みのデータを入れていきます。

最後に「putImageData()」メソッドを使ってCanvasに描画を行います。これでCanvasにグレースケールに変換した画像が描画されます。

COLUMN

ローカルディスク上での動作について

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります(ブラウザによるが、セキュリティエラーになることがある)。

COLUMN　処理する画像が大きい場合

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

COLUMN　Safari 6とRetina Displayの場合

Safari 6が高解像度のRetina Display上で動作している場合、サンプルを実行するとピクセルが右下にずれることがあります(0.5pxほど、ずれる)。

関連項目 ▶▶▶

- エンボス加工する …… p.484
- 画像をぼかす …… p.488
- 画像のガンマ補正を行う …… p.492
- 画像を白黒化(2値化)する …… p.495
- 画像を特定の色相(トーン)にする …… p.499
- 要素にグレーフィルタを適用する …… p.954

SECTION-211

エンボス加工する

ここでは、Canvasに表示されているピクセルを操作しエンボス加工する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="エンボス加工" id="effect">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

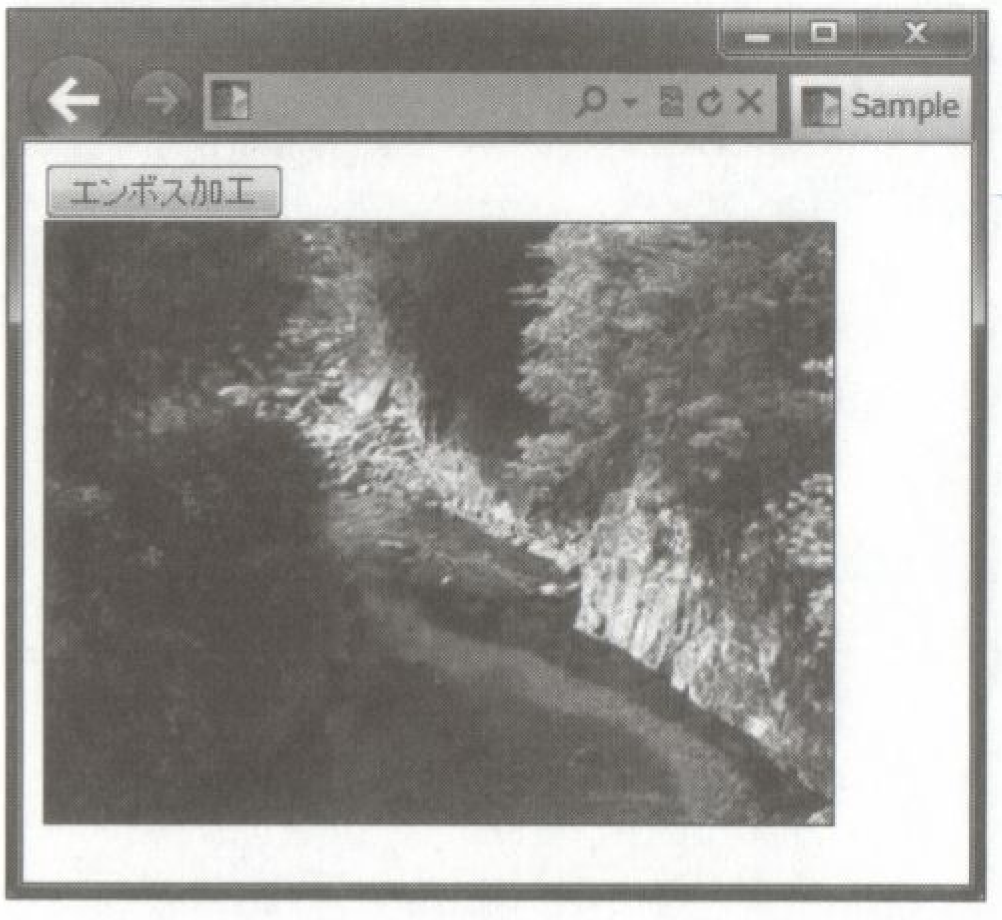
SAMPLE CODE JavaScriptのコード(sample.js)

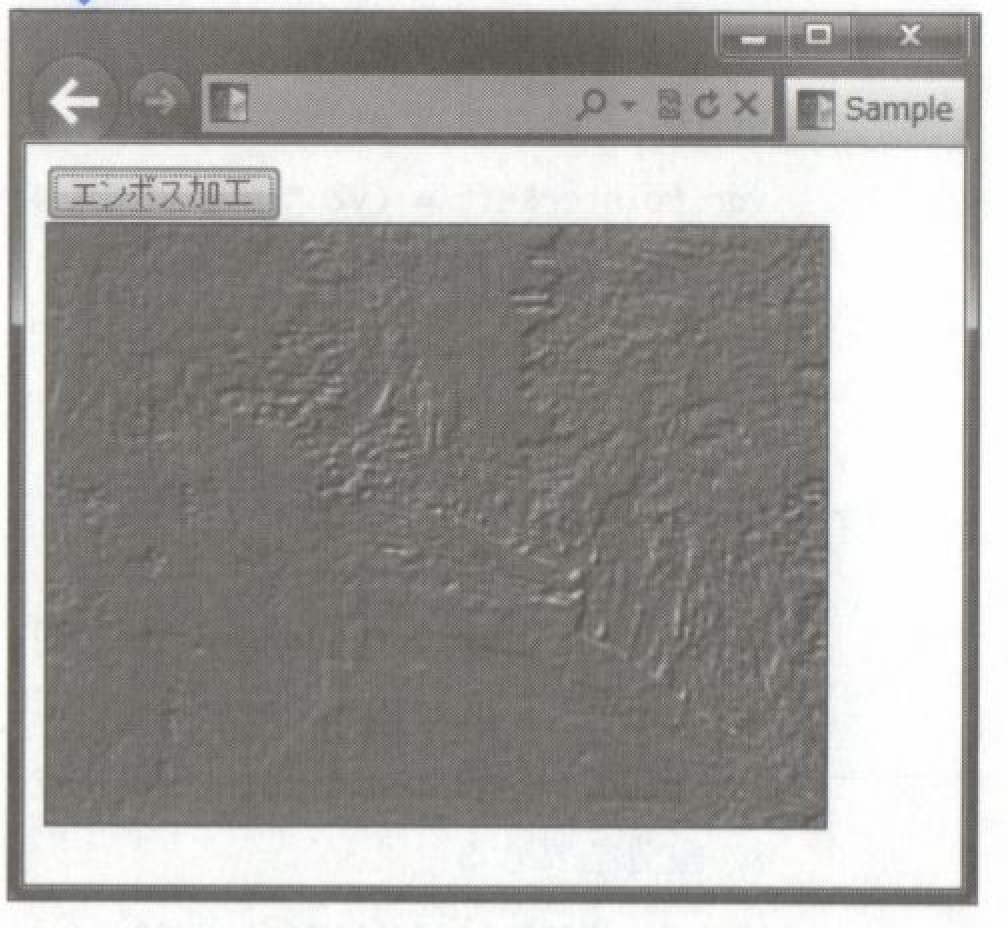
```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // エンボスに変換する関数を呼び出す
    toEmboss(context, myCanvas.width, myCanvas.height);
  }, false);
  // エンボスに加工する関数
  function toEmboss(context, w, h){
    // Canvas内のピクセルデータを読み出す
    var pixelData = context.getImageData(0,0, w, h).data;
```

```
  // 縦方向のピクセル数だけ繰り返す
  for(var y=h-1; y>=0; y--){
   // 横方向のピクセル数だけ繰り返す
    for(var x=w-1; x>=0; x--){
      // 処理するピクセルの配列内での位置を計算する
      // 配列要素は「赤」「緑」「青」「不透明度(α)」の順番
      var pointer = (y * w + x) * 4;
      // 差分を取るピクセルの位置を設定する
      var x2 = x - 2;
      var y2 = y - 2;
      // 画面外にならないように調整する
      if (x2 < 0){ x2 = 0; }
      if (y2 < 0){ y2 = 0; }
      var pointerLeft = (y2 * w + x2) * 4;
      // 色データを読み出し
      var red = pixelData[pointer+0];
      var green = pixelData[pointer+1];
      var blue = pixelData[pointer+2];
      var alpha = pixelData[pointer+3];
      var red2 = pixelData[pointerLeft+0];
      var green2 = pixelData[pointerLeft+1];
      var blue2 = pixelData[pointerLeft+2];
      // 差分を求める
      var r = Math.floor((red2 - red)/2)+128;
      var g = Math.floor((green2 - green)/2)+128;
      var b = Math.floor((blue2 - blue)/2)+128;
      // グレーにしたい場合は以下のコード
      // var gray = green*0.6+red*0.3+blue*0.1;
      // var gray2 = green2*0.6+red2*0.3+blue2*0.1;
      // var r = g = b = Math.floor((gray2 - gray)/2)+128;
      // 処理結果を配列に入れる
      pixelData[pointer+0] = r;
      pixelData[pointer+1] = g;
      pixelData[pointer+2] = b;
      pixelData[pointer+3] = alpha;
    }
  }
  // 処理結果をCanvasに描画するために新規にピクセルデータを生成
  var output = context.createImageData(w, h);
  // ピクセルデータの総数だけ繰り返す
  for(var i=0; i<w*h*4; i++){
   // 操作したピクセルを入れる
    output.data[i] = pixelData[i];
  }
  // Canvasに描画する
  context.putImageData(output, 0,0);
}
```

```
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、Canvasに描画されている画像がエンボス加工されて表示されます。





ONEPOINT エンボス加工するには2つのピクセルの輝度の差分を取る

Canvasに描画されている画像を加工してエンボスにするには、2つのピクセルの輝度の差分を取ります。差分を2で割った後に、オフセット値を加算します。Canvasでは「0」～「255」までの輝度の範囲なので、中間の値である「128」を加算します。

また、完全なグレースケールでのエンボス加工を行う場合には、2つのピクセルの輝度からグレースケール値を求め、その差分を取ります。求めた値を2で割り、128を加算すればグレースケールでのエンボス画像になります。

なお、ピクセル操作に関しては、480ページを参照してください。

COLUMN ローカルディスク上での動作について

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります（ブラウザによるが、セキュリティエラーになることがある）。

COLUMN **処理する画像が大きい場合**

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

関連項目 ▶▶▶

- グレースケール画像にする ………… p.480
- 画像をぼかす ………… p.488
- 画像のガンマ補正を行う ………… p.492
- 画像を白黒化(2値化)する ………… p.495
- 画像を特定の色相(トーン)にする ………… p.499

SECTION-212

画像をぼかす

ここでは、Canvasに表示されている画像をぼかす方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form><input type="button" value="ぼかし" id="effect"></form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

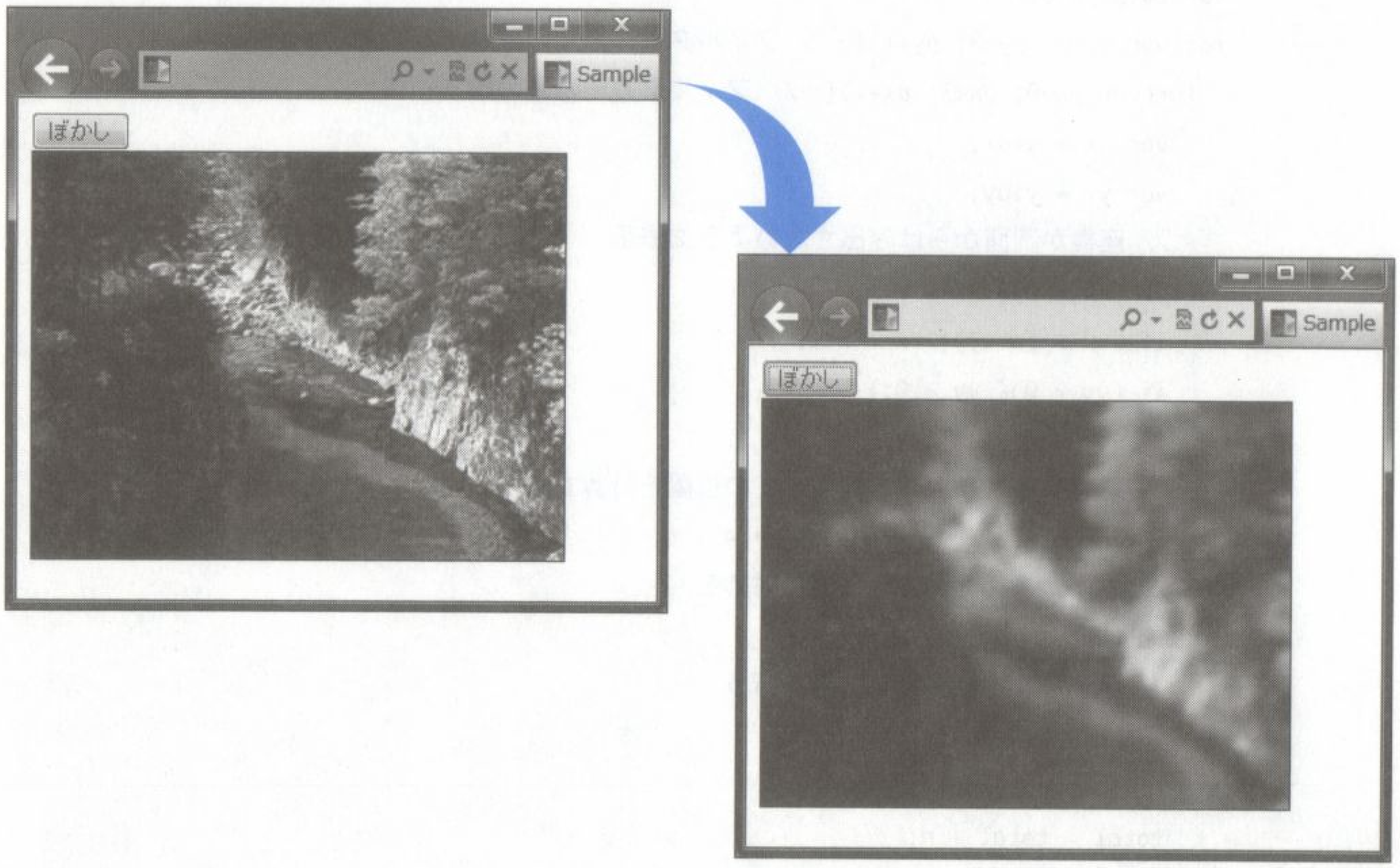
```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // ぼかし処理を行う関数を呼び出す
    blur(context, myCanvas.width, myCanvas.height,
      [ 1/9, 1/9, 1/9,
        1/9, 1/9, 1/9,
        1/9, 1/9, 1/9 ]
    );
  }, false);
  // フィルタ処理関数
  function blur(context, w, h, pat){
    // Canvas内のピクセルデータを読み出す
    var pixelData = context.getImageData(0,0, w, h).data;
```

```
// 縦方向のピクセル数だけ繰り返す
for(var y=h; y>=-1; y--){
  // 横方向のピクセル数だけ繰り返す
  for(var x=w; x>=-1; x--){
    for(var c=0; c<3; c++){
      var total = 0;
      for(var py=0; py<3; py++){  // フィルタカーネルの縦サイズ
        for(var px=0; px<3; px++){  // フィルタカーネルの横サイズ
          var xx = x+px;
          var yy = y+py;
          // 座標が画面からはみ出さないように修正
          if (xx < 0){ xx = 0;}
          if (xx > (w-1)){ xx = w-1; }
          if (yy < 0){ yy = 0;}
          if (yy > (h-1)){ yy = h-1; }
          // 処理するピクセルの配列内での位置を計算する
          var pointer = (yy * w + xx) * 4 + c;
          // フィルタカーネルの位置を求める
          var kpos = py * 3 + px;
          // 色データを読み出し計算を行う
          var n = pixelData[pointer];
          n = n * pat[kpos];
          total = total + n;
        }
      }
      // 処理したピクセルを格納する位置を計算
      var xx = x+1;
      var yy = y+1;
      // 座標が画面からはみ出さないように修正
      if (xx < 0){ xx = 0;}
      if (xx > (w-1)){ xx = w-1; }
      if (yy < 0){ yy = 0;}
      if (yy > (h-1)){ yy = h-1; }
      // 処理結果を配列に入れる
      pointer = (yy * w + xx) * 4 + c;
      pixelData[pointer] = total;
    }
  }
}
// 処理結果をCanvasに描画するために新規にピクセルデータを生成
var output = context.createImageData(w, h);
// ピクセルデータの総数だけ繰り返す
for(var i=0; i<w*h*4; i++){
  // 操作したピクセルを入れる
  output.data[i] = pixelData[i];
}
context.putImageData(output, 0,0);  // Canvasに描画する
```

01 02 03 04 05 06 07 08 09 10 CHAPTER Canvas/Canvas 3D/SVG 11 12 13 14

```
    }
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、Canvasに描画されている画像がぼやけます。何度もボタンをクリックすると、より画像がぼけていきます。



ONEPOINT 画像をぼかすにはピクセルの平均を取る

Canvasに描画されている画像をぼかすには、ピクセルの平均を取ります（平滑化フィルタ/平均化フィルタ）。中心の周辺にあるピクセルとの輝度の平均をとることで画像がぼやけていきます。プログラム中の以下の部分の数値を変更することで、ぼかし具合を設定することができます。なお、ピクセル操作に関しては、480ページを参照してください。

```
[ 1/9, 1/9, 1/9,
  1/9, 1/9, 1/9,
  1/9, 1/9, 1/9 ]
```

COLUMN ローカルディスク上での動作について

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります（ブラウザによるが、セキュリティエラーになることがある）。

COLUMN 処理する画像が大きい場合

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

COLUMN ガウスぼかしを適用するには

平滑化フィルタではなく、ガウスぼかしにするには、次のように配列の値（カーネル値）を設定します。

```
blur(context, myCanvas.width, myCanvas.height,
  [ 1/16, 2/16, 1/16,
    2/16, 4/16, 2/16,
    1/16, 2/16, 1/16 ]
);
```

関連項目 ▶▶▶

- グレースケール画像にする ……………… p.480
- エンボス加工する ……………… p.484
- 画像のガンマ補正を行う ……………… p.492
- 画像を白黒化（2値化）する ……………… p.495
- 画像を特定の色相（トーン）にする ……………… p.499
- 要素にぼかしフィルタを適用する ……………… p.971

SECTION-213

画像のガンマ補正を行う

ここでは、Canvasに表示されている画像のガンマ補正を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="text" value="1.8" id="gamma">
      <input type="button" value="ガンマ補正" id="effect">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // Canvasに画像を描画
    context.drawImage(imgObj, 0, 0);
    // 入力されたガンマ値を読み出す
    var gValue = parseFloat(document.getElementById("gamma").value);
    // ガンマ補正処理を行う関数を呼び出す
    gammaAdjust(context, myCanvas.width, myCanvas.height, gValue);
  }, false);
```

```
// ガンマ補正処理関数
function gammaAdjust(context, w, h, gamma){
  // Canvas内のピクセルデータを読み出す
  var pixelData = context.getImageData(0,0, w, h).data;
  // 縦方向のピクセル数だけ繰り返す
  for(var y=0; y<h; y++){
    // 横方向のピクセル数だけ繰り返す
    for(var x=0; x<w; x++){
      // 処理するピクセルの配列内での位置を計算する
      // 配列要素は「赤」「緑」「青」「不透明度(α)」の順番
      var pointer = (y * w + x) * 4;
      // 色データを読み出し
      var red = pixelData[pointer+0];
      var green = pixelData[pointer+1];
      var blue = pixelData[pointer+2];
      var alpha = pixelData[pointer+3];
      // ガンマ補正
      var red = 255 * Math.pow((red / 255), 1/gamma);
      var green = 255 * Math.pow((green / 255), 1/gamma);
      var blue = 255 * Math.pow((blue / 255), 1/gamma);
      // 補正結果を配列に入れる
      pixelData[pointer+0] = red;
      pixelData[pointer+1] = green;
      pixelData[pointer+2] = blue;
      pixelData[pointer+3] = alpha;
    }
  }
  // 処理結果をCanvasに描画するために新規にピクセルデータを生成
  var output = context.createImageData(w, h);
  // ピクセルデータの総数だけ繰り返す
  for(var i=0; i<w*h*4; i++){
    // 操作したピクセルを入れる
    output.data[i] = pixelData[i];
  }
  // Canvasに描画する
  context.putImageData(output, 0,0);
 }
}, false);
```

ブラウザでindex.htmlを表示し、ガンマ値を入力しボタンをクリックすると、Canvasに描画されている画像に対してガンマ補正が行われます。



ONEPOINT **画像のガンマ補正を行うには「255×(輝度÷255) $^{1/\gamma}$」の計算式を使う**

Canvasに描画されている画像のガンマ補正を行うには、「255×(輝度÷255) $^{1/\gamma}$」の計算式を使います。すべてのピクセル値を読み出し、この計算式に従って求めた値をCanvasに書き戻します。

なお、ピクセル操作に関しては、480ページを参照してください。

COLUMN **ローカルディスク上での動作のついて**

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります(ブラウザによるが、セキュリティエラーになることがある)。

COLUMN **処理する画像が大きい場合**

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

関連項目 ▶▶▶

● グレースケール画像にする ……… p.480
● エンボス加工する ……… p.484
● 画像をぼかす ……… p.488
● 画像を白黒化(2値化)する ……… p.495
● 画像を特定の色相(トーン)にする ……… p.499
● 要素の輝度(明るさ)を変更する ……… p.962

SECTION-214

画像を白黒化(2値化)する

ここでは、Canvasに表示されている画像を白黒化(2値化)する方法について解説します。

SAMPLE CODE HTML(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      しきい値:<input type="text" value="127" id="threshold">
      <input type="button" value="白黒にする" id="effect">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

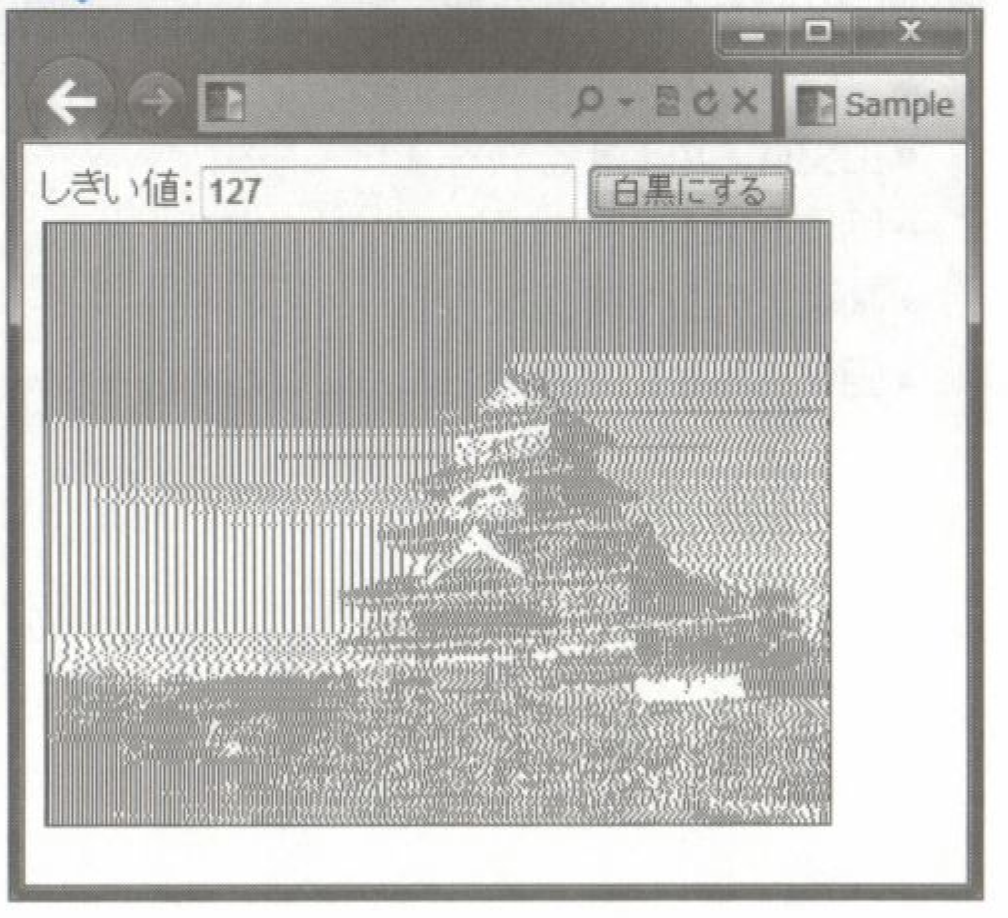
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // 画像を描画
    context.drawImage(imgObj, 0, 0);
    // しきい値を読み出す
    var n = parseInt(document.getElementById("threshold").value);
    // 白黒化(2値化)を行う関数を呼び出す
    mono(context, myCanvas.width, myCanvas.height, n);
  }, false);
```

```
  // 白黒化(2値化)処理関数(誤差拡散)
  function mono(context, w, h, threshold){
    // Canvas内のピクセルデータを読み出す
    var pixelData = context.getImageData(0,0, w, h).data;
    // 縦方向のピクセル数だけ繰り返す
    for(var y=0; y<h; y++){
      // 誤差を格納する変数
      var diff = 0;
      // 横方向のピクセル数だけ繰り返す
      for(var x=0; x<w; x++){
        // 処理するピクセルの配列内での位置を計算する
        // 配列要素は「赤」「緑」「青」「不透明度(α)」の順番
        var pointer = (y * w + x) * 4;
        // 色データを読み出し
        var red = pixelData[pointer+0];
        var green = pixelData[pointer+1];
        var blue = pixelData[pointer+2];
        var alpha = pixelData[pointer+3];
        // 輝度を計算
        var gray = green * 0.6 + red * 0.3 + blue * 0.1;
        gray = gray + diff;
        // しきい値より大きいか小さいか
        if (gray > threshold){
          diff = gray - 255;
          gray = 255;  // 白にする
        }else{
          diff = 255 - gray;
          gray = 0;
        }
        // 補正結果を配列に入れる
        pixelData[pointer+0] = gray;
        pixelData[pointer+1] = gray;
        pixelData[pointer+2] = gray;
        pixelData[pointer+3] = alpha;
      }
    }
    // 処理結果をCanvasに描画するために新規にピクセルデータを生成
    var output = context.createImageData(w, h);
    // ピクセルデータの総数だけ繰り返す
    for(var i=0; i<w*h*4; i++){
      // 操作したピクセルを入れる
      output.data[i] = pixelData[i];
    }
    // Canvasに描画する
    context.putImageData(output, 0,0);
  }
}, false);
```

ブラウザでindex.htmlを表示し、しきい値を入力してボタンをクリックすると、Canvasに描画されている画像が白黒に変換されます。





ONEPOINT

画像を白黒化(2値化)するには
しきい値によって黒と白にピクセルを置き換える

Canvasに描画されている画像を白黒化(2値化)するには、しきい値よりも小さい場合は黒に、大きい場合は白のピクセルに置き換えていきます。ただし、この方法では画像の情報が失われます。そこで、失われた情報を次のピクセルに渡すようにします(誤差分散/誤差拡散)。渡された情報(輝度の差分)を加算し、しきい値と比較して白黒のピクセルに置き換えていきます。これを画素数分行えば、Canvasの画像を白黒化(2値化)することができます。

なお、ピクセル操作に関しては、480ページを参照してください。

COLUMN

ローカルディスク上での動作について

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります(ブラウザによるが、セキュリティエラーになることがある)。

COLUMN 処理する画像が大きい場合

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

関連項目 ▶▶▶

- グレースケール画像にする …… p.480
- エンボス加工する …… p.484
- 画像をぼかす …… p.488
- 画像のガンマ補正を行う …… p.492
- 画像を特定の色相(トーン)にする …… p.499

SECTION-215

画像を特定の色相(トーン)にする

ここでは、Canvasに表示されている画像を特定の色相(トーン)にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      色相：<input type="text" value="60" id="hueValue">
      <input type="button" value="指定したトーンにする" id="effect">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

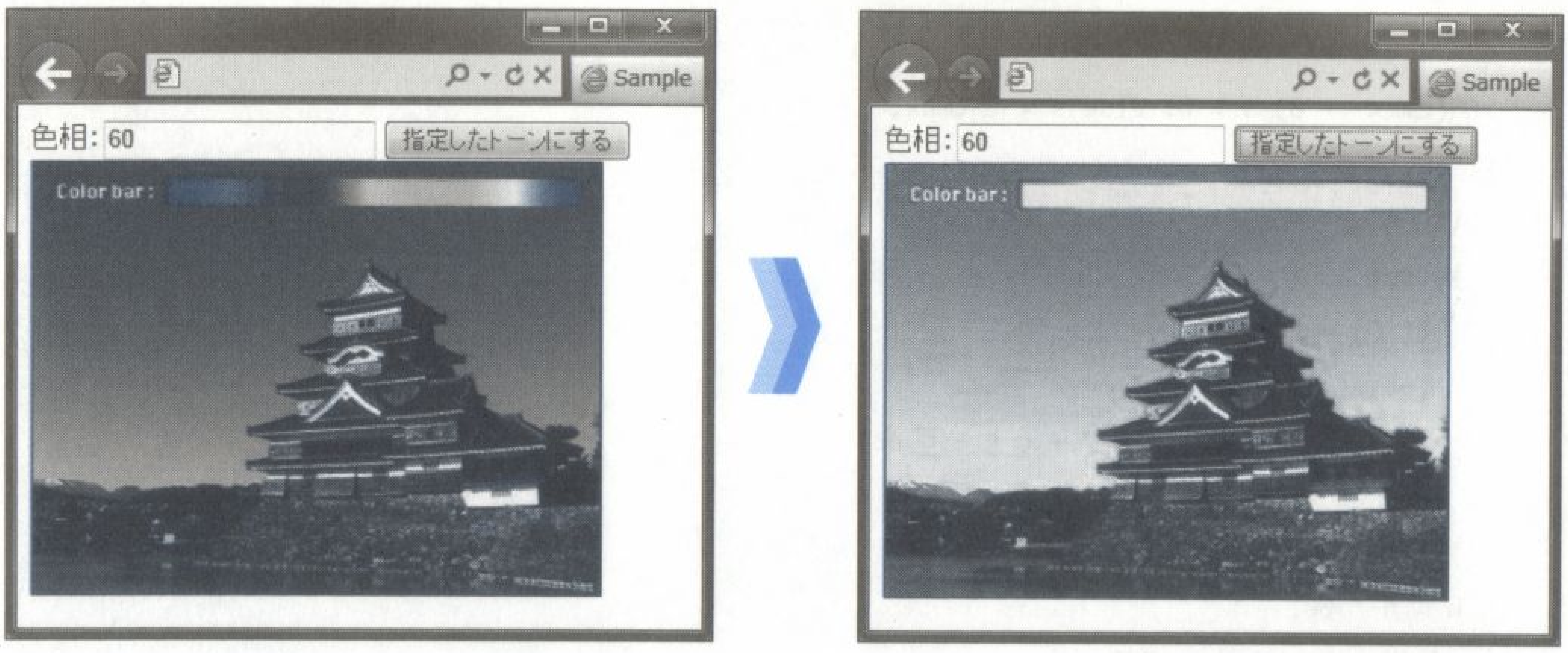
```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
  }
  // ボタンにイベントを割り当て
  document.getElementById("effect").addEventListener("click", function(){
    // 画像を描画
    context.drawImage(imgObj, 0, 0);
    // 色相の値を読み出す
    var n = parseFloat(document.getElementById("hueValue").value);
    // 特定の色相で塗り潰す処理を行う関数を呼び出す
    colorTone(context, myCanvas.width, myCanvas.height, n);
```

▼

```
  }, false);
  // 白黒化(2値化)処理関数(誤差拡散)
  function colorTone(context, w, h, newTone){
    // Canvas内のピクセルデータを読み出す
    var pixelData = context.getImageData(0,0, w, h).data;
    // 縦方向のピクセル数だけ繰り返す
    for(var y=0; y<h; y++){
      // 横方向のピクセル数だけ繰り返す
      for(var x=0; x<w; x++){
        // 処理するピクセルの配列内での位置を計算する
        // 配列要素は「赤」「緑」「青」「不透明度(α)」の順番
        var pointer = (y * w + x) * 4;
        // 色データを読み出し
        var red = pixelData[pointer+0];
        var green = pixelData[pointer+1];
        var blue = pixelData[pointer+2];
        var alpha = pixelData[pointer+3];
        // RGBからHSVに変換
        var hsv = RGBtoHSV(red, green, blue);
        // 指定した色相にした後、再度RGBに変換する
        var rgb = HSVtoRGB(newTone, hsv.S, hsv.V);
        // 補正結果を配列に入れる
        pixelData[pointer+0] = rgb.R*255;
        pixelData[pointer+1] = rgb.G*255;
        pixelData[pointer+2] = rgb.B*255;
        pixelData[pointer+3] = alpha;
      }
    }
    // 処理結果をCanvasに描画するために新規にピクセルデータを生成
    var output = context.createImageData(w, h);
    // ピクセルデータの総数だけ繰り返す
    for(var i=0; i<w*h*4; i++){
      // 操作したピクセルを入れる
      output.data[i] = pixelData[i];
    }
    // Canvasに描画する
    context.putImageData(output, 0,0);
  }
}, false);
// RGBからHSV (R:0~255, G:0~255, B:0~255, h:0~360, s:0~1, B:0~1)
function RGBtoHSV(r, g, b){
  var h = s = v = 0;
  if ( r >= g)   cmax = r; else cmax = g;
  if ( b > cmax) cmax = b;
  if ( r <= g)   cmin = r; else cmin = g;
  if ( b < cmin) cmin = b;
  v = cmax;
```

```
    var c = cmax - cmin;
    if (cmax == 0) s = 0; else s = c/cmax;
    if (s != 0){
      if (r == cmax){ h = (g - b)/c;
      }else{
        if (g == cmax){ h = 2 + (b - r)/c;
        }else{
          if (b == cmax) h = 4 + ( r - g)/c;
        }
      }
      h = h * 60;
      if (h < 0) h = h + 360;
    }
    return { H:h, S:s, V:v/255 }
  }
  // HSVからRGB (R:0〜1, G:0〜1, B:0〜1, h:0〜360, s:0〜1, B:0〜1)
  function HSVtoRGB(h, s, v){
    var r = g = b = 0;
    if (s < 0) s = 0;
    if (s > 1) s = 1;
    if (v < 0) v = 0;
    if (v > 1) v = 1;
    h = h % 360;
    if (h < 0) h = h + 360;
    h = h / 60;
    i = Math.floor(h);
    var f = h - i;
    p1 = v * (1 - s);
    p2 = v * (1 - s*f);
    p3 = v * (1 - s*(1 - f));
    if (i == 0) { r = v; g = p3; b = p1; }
    if (i == 1) { r = p2;g = v;  b = p1; }
    if (i == 2) { r = p1;g = v;  b = p3; }
    if (i == 3) { r = p1;g = p2; b = v;  }
    if (i == 4) { r = p3;g = p1; b = v;  }
    if (i == 5) { r = v; g = p1; b = p2; }
    return { R:r, G:g, B:b }
  }
```

ブラウザでindex.htmlを表示し、色相の値(0〜360)を入力してボタンをクリックすると、指定した色相(トーン)の画像に変換されます。



ONEPOINT 画像を特定の色相(トーン)にするには ピクセルの色情報をRGBからHSVやHSLなどに変換する

Canvasに描画されている画像を特定の色相(トーン)にするには、ピクセルの色情報をRGBからHSVやHSLなどに変換します。HSV/HSLに変換すると、色情報は色相・彩度・明度/輝度になります。色相だけを変えて再度、RGBに変換することで画像の色相(トーン)だけを変えることができます。サイドや輝度なども調整することでセピア調などにすることもできます。なお、ピクセル操作に関しては、480ページを参照してください。

COLUMN ローカルディスク上での動作について

ローカルディスク上では動作しないブラウザがあります。また、サーバー上にある画像は処理できますが、ローカルディスク上にある画像をCanvasに描画した場合、その画像のピクセルデータは読み出せないことがあります(ブラウザによるが、セキュリティエラーになることがある)。

COLUMN 処理する画像が大きい場合

大きいサイズの画像を処理する場合、処理に時間がかかり、ブラウザが何も操作できなくなってしまう場合があります。このような場合は、並列処理を行うWeb Workersを使用します。なお、Android 2、Windows Phoneの場合は、UI操作を行う処理は別に動作するため、画像処理中でもブラウザの操作を行うことができます。

関連項目 ▶▶▶

● グレースケール画像にする …… p.480
● エンボス加工する …… p.484
● 画像をぼかす …… p.488
● 画像のガンマ補正を行う …… p.492
● 画像を白黒化(2値化)する …… p.495
● 要素にトーン(色相変換)フィルタを適用する …… p.958

SECTION-216

パターンを生成する

ここでは、パターンを生成し、生成したパターンで図形を塗りつぶす方法について解説します。

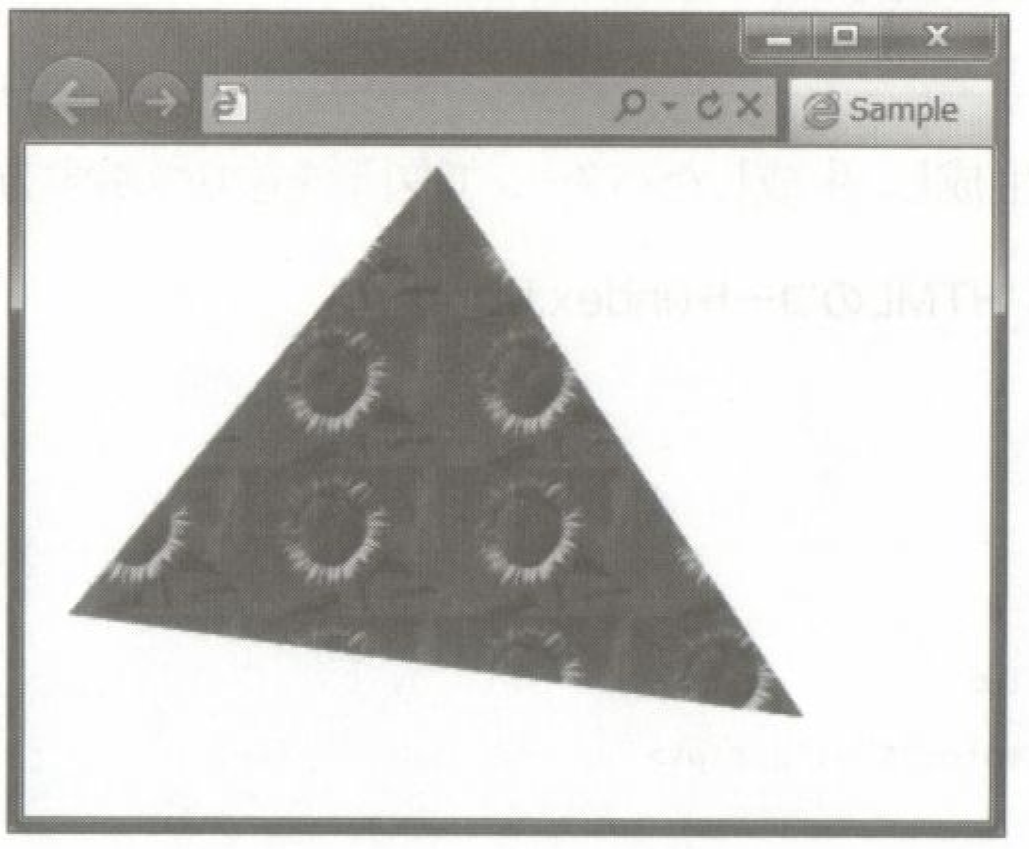
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 画像を描画する
  var imgObj = new Image();
  imgObj.src = "images/sunflower.jpg";
  imgObj.onload = function(){
    // 画像をパターンとして登録
    var pattern1 = context.createPattern(this, "repeat");
    // 塗りつぶしをパターンにする
    context.fillStyle = pattern1;
    // 三角形を描画
    context.beginPath();
    context.moveTo(160,0);
    context.lineTo(310, 220);
    context.lineTo(10, 180);
    context.closePath();
    // パターンで塗りつぶす
    context.fill();
  }
}, false);
```

ブラウザでindex.htmlを表示すると、三角形が表示され、その中が指定したパターンで塗りつぶされます。



ONEPOINT　パターンを生成するには「createPattern()」メソッドを使う

パターンを生成するには、「createPattern()」メソッドを使います。「createPattern()」は、画像をパターンにするメソッドです。このため、最初のパラメータには、画像オブジェクトを指定します。2番目のパラメータには、次の表の文字列が指定できます。2番目のパラメータが空文字などの場合は、「repeat」を指定したのと同じ結果になります。

文字列	説明
repeat	縦横方向に繰り返し
repeat-x	横方向に繰り返し
repeat-y	縦方向に繰り返し

「createPattern()」メソッドは、生成したパターンを返します。生成したパターンを利用するには、「fillStyle」プロパティに代入します。これにより、図形で「fill()」メソッドを実行すると、図形の内部がパターンで描画されます。

SECTION-217

影を描画する

ここでは、影を描画する方法について解説します。

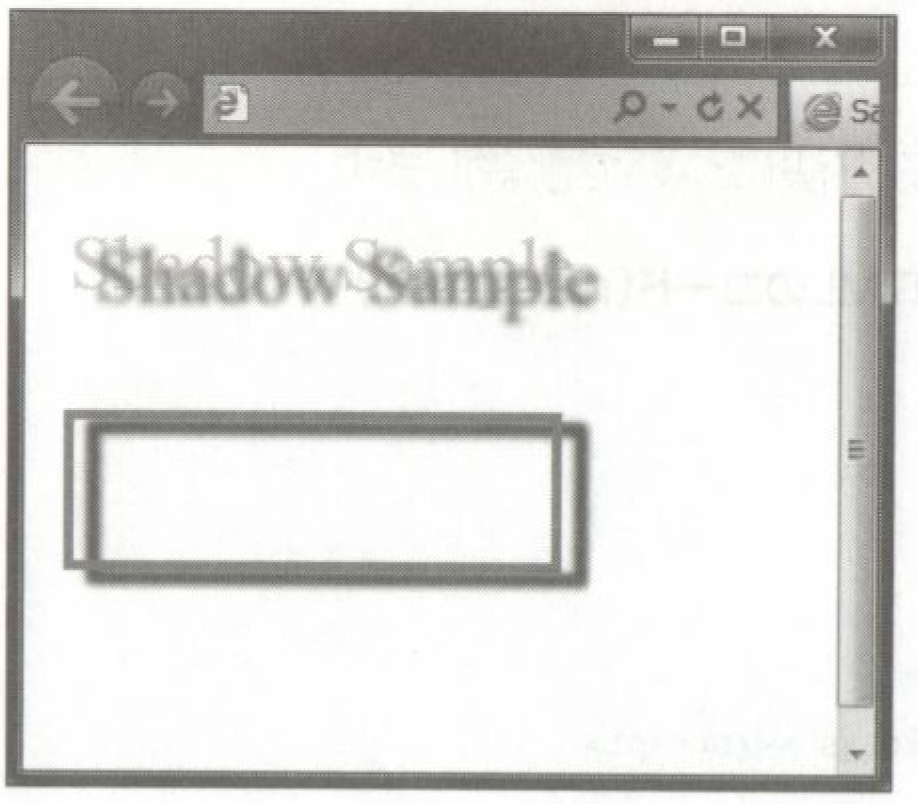
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // フォントとサイズ、色を設定する
  context.font = "24pt 'Tahoma-Bold'";
  context.fillStyle = "rgba(0, 255, 0, 0.5)";
  // 影の設定を行う
  context.shadowColor = "blue";
  context.shadowBlur = 3;
  context.shadowOffsetX = 10;
  context.shadowOffsetY = 5;
  // 文字を描画する
  context.fillText("Shadow Sample", 10, 50);
  // 四角形を描画する
  context.strokeStyle = "red";
  context.lineWidth = 4;
  context.strokeRect(10, 100, 200, 60);
}, false);
```

ブラウザでindex.htmlを表示すると、文字と四角形に影が描画されます。描画される図形に不透明度が指定されている場合、影にもその不透明度が反映されます。



ONEPOINT

影を描画するには「shadowColor」「shadowBlur」「shadowOffsetX」「shadowOffsetY」プロパティを使う

影を描画するには、次の表に示すプロパティを設定します。

プロパティ	説明
shadowColor	影の色
shadowBlur	影のぼかし具合
shadowOffsetX	影の横のずれ(オフセット)
shadowOffsetY	影の縦のずれ(オフセット)

「shadowColor」プロパティは、カラー名やカラーコードなど、CSS3で指定できるカラーを指定することができます。「shadowBlur」プロパティは影のぼかし具合を指定し、値が大きくなるほどぼかし具合が強くなります。影のずれ具合は、「shadowOffsetX」プロパティと「shadowOffsetY」プロパティで指定します。

なお、影は、描画する図形や文字の不透明度を引き継いで描画されます。

COLUMN

Androidの「shadowOffsetY」プロパティの不具合

Android 2〜3の標準ブラウザでは、「shadowOffsetY」プロパティで指定した方向が仕様とは逆方向になります。通常、正数を指定すると下方向に影が描画されますが、Android 2〜3の標準ブラウザでは上方向に影が描画されてしまいます。なお、Android 4では修正されています。

SECTION-218

映像をCanvasに描画する

ここでは、映像をCanvasに描画する方法について解説します。

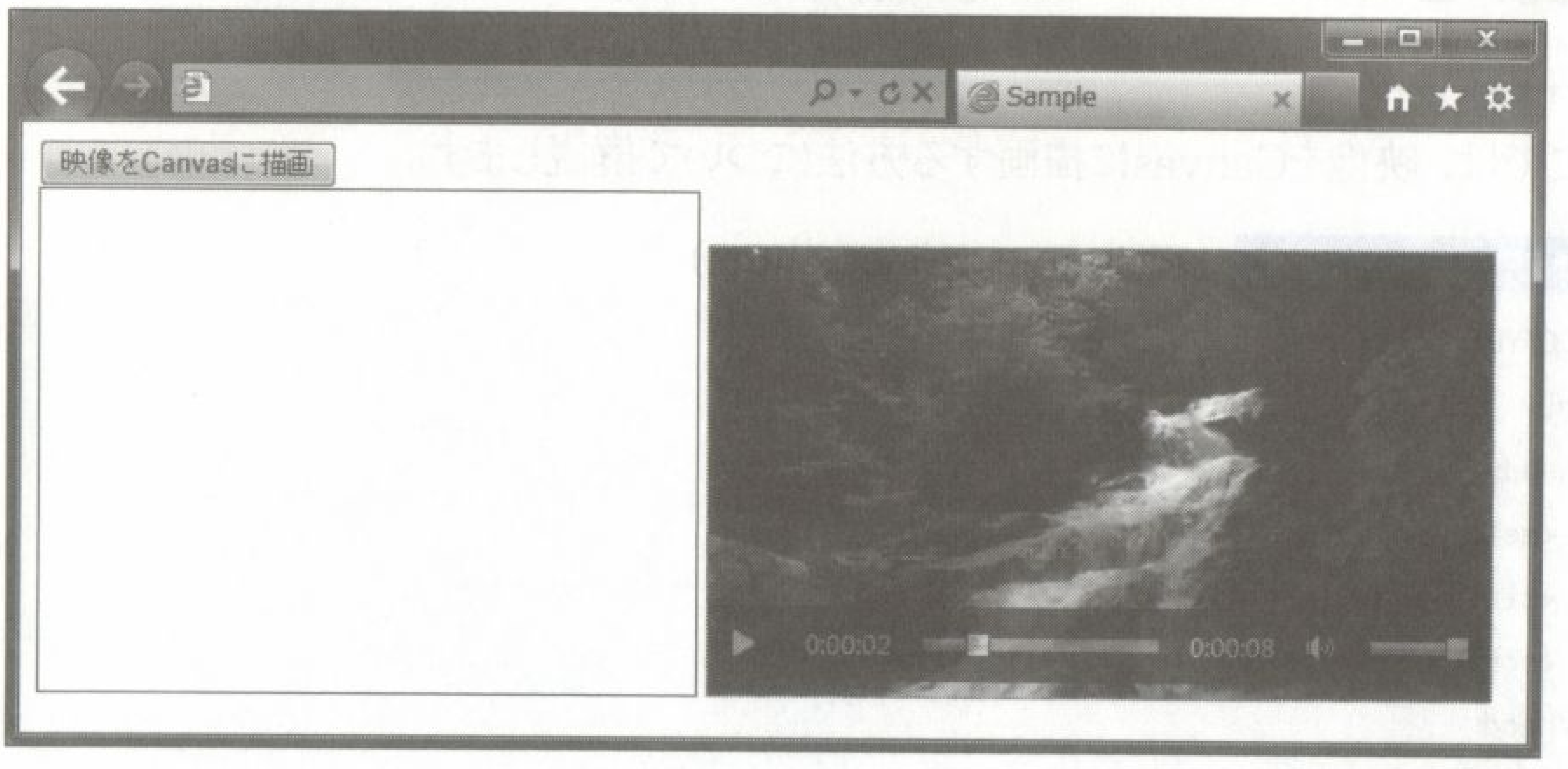
SAMPLE CODE HTMLのコード(index.html)

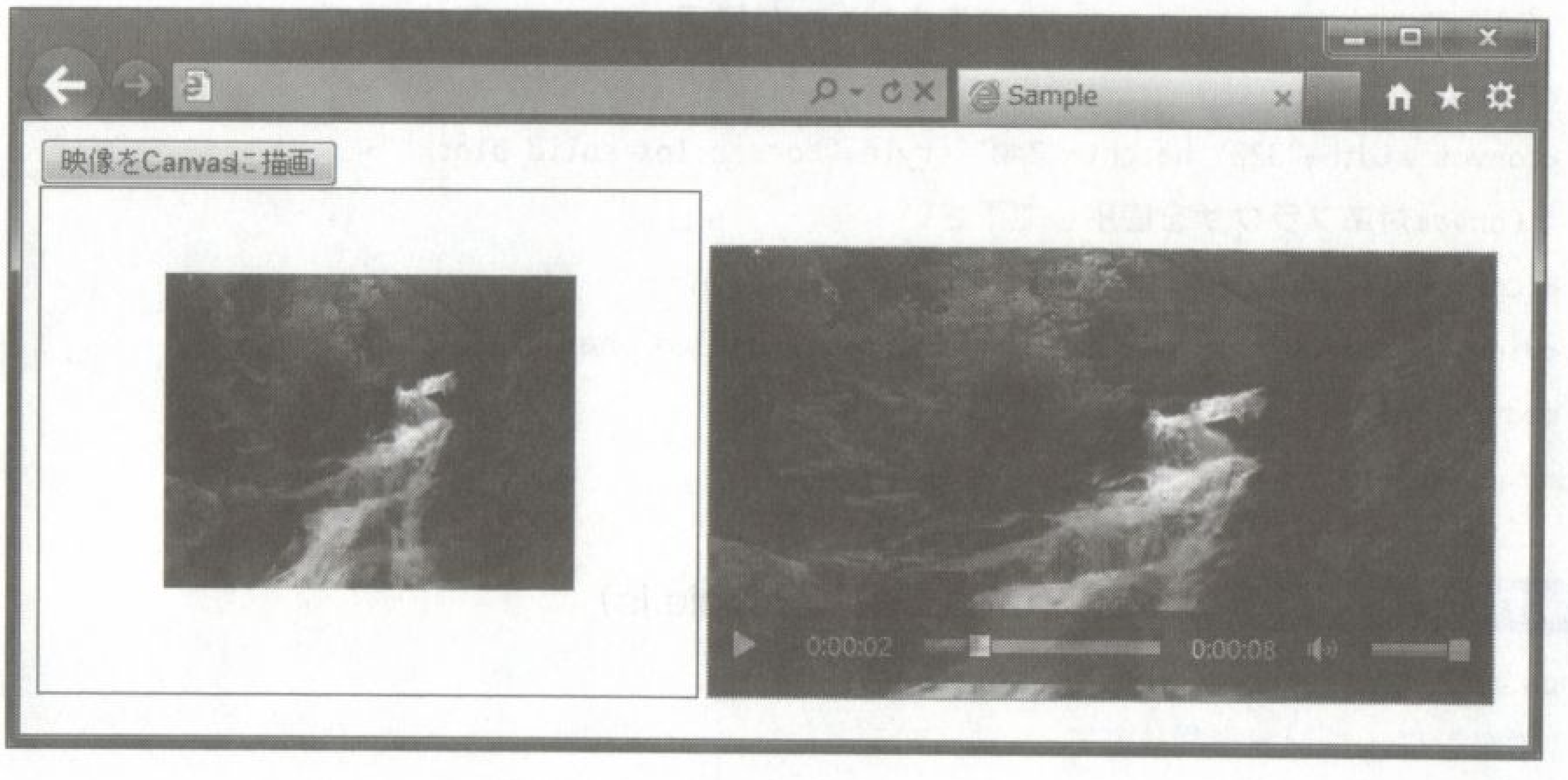
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="映像をCanvasに描画" id="draw">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
    <video src="movie/sample.mp4" controls width="384" height="216"></video>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを割り当て
  document.getElementById("draw").addEventListener("click", function(){
    // Video要素を読み出し
    var myVideo = document.getElementsByTagName("video")[0];
    // Canvas要素を読み出し
    var myCanvas = document.getElementsByTagName("canvas")[0];
    // コンテキストを取得
    var context = myCanvas.getContext("2d");
    if (!context){ return; }
    // 映像を小さくして描画する
    context.drawImage(myVideo, 60, 40, 200, 150);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「映像をCanvasに描画」ボタンをクリックすると、現在再生されている映像がCanvasに描画されます。





ONEPOINT **映像を描画するには「drawImage()」メソッドを使う**

Canvasに映像を描画するには、「drawImage()」メソッドを使います。「drawImage()」メソッドの最初のパラメータに「video」要素を指定します。これでCanvasに「video」要素で表示されている映像が描画されます。「drawImage()」メソッドの2番目以降のパラメータは画像を描画する場合と同じで、次のようになります。

◉座標を指定して元のサイズで描画

```
drawImage(video, x, y)
```

◉座標とサイズを指定して描画

```
drawImage(video, x, y, width, height)
```

◉元映像の一部をクリッピングした後で指定した座標とサイズで描画

```
drawImage(video, sourceX, sourceY, sourceWidth, sourceHeight, destinationX, destinationY,
 destinationWidth, destinationY)
```

SECTION-219

Canvasの内容を他のCanvasに描画する

ここでは、Canvasの内容を他のCanvasに描画する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

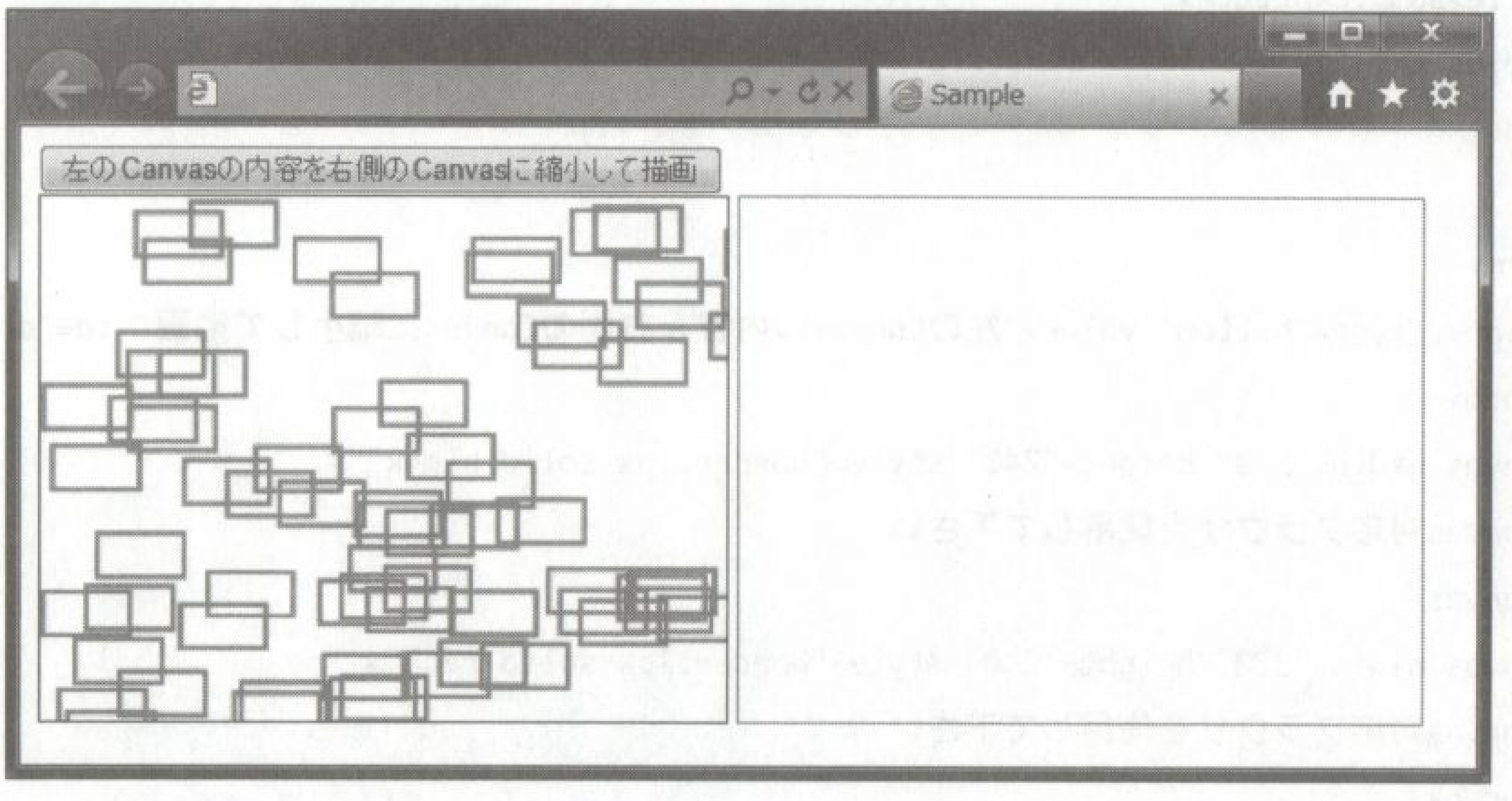
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="左のCanvasの内容を右側のCanvasに縮小して描画" id="draw">
    </form>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
    <canvas width="320" height="240" style="border:1px solid red"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

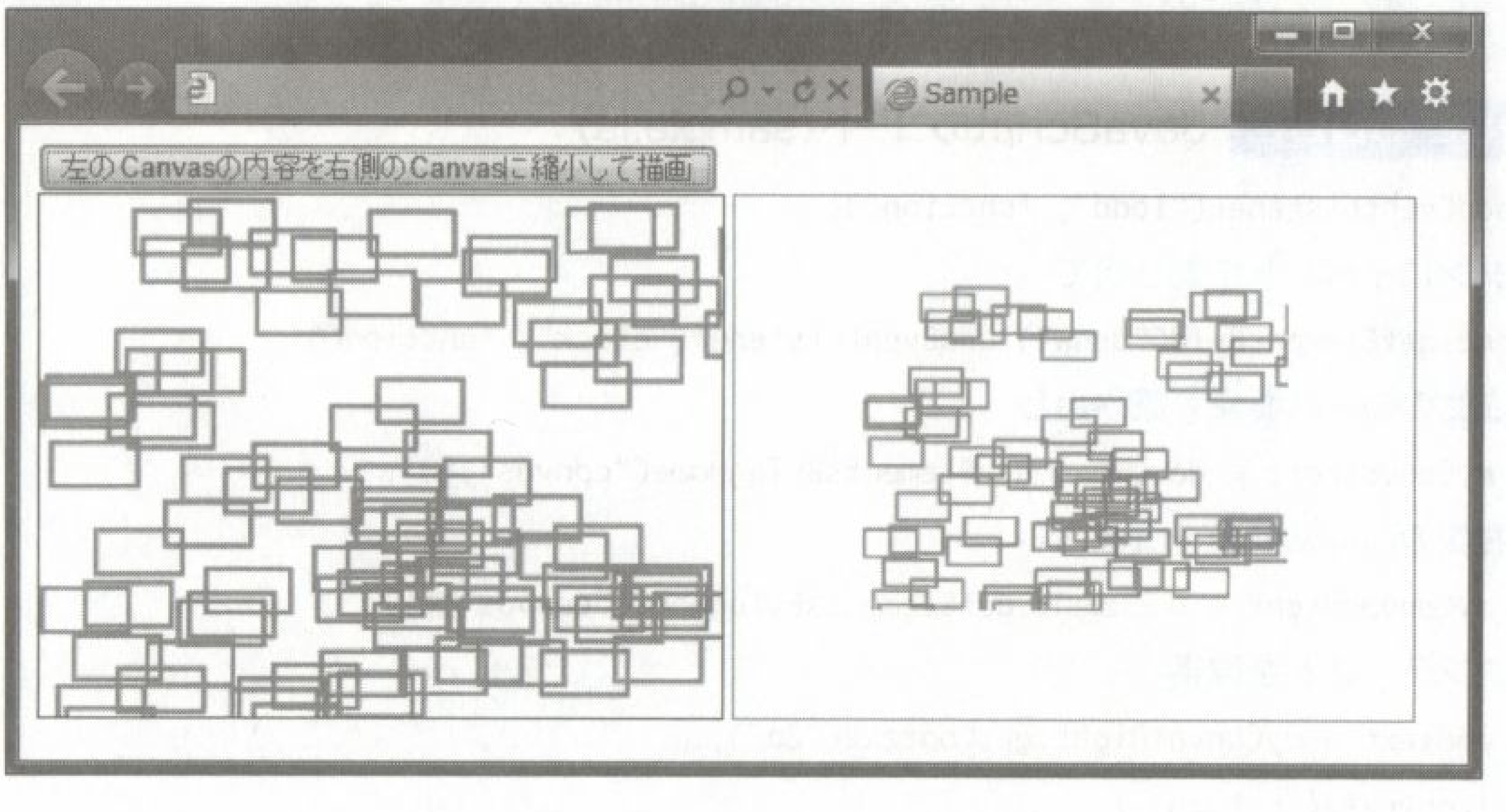
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを割り当て
  document.getElementById("draw").addEventListener("click", function(){
    // 左側のCanvas要素を読み出し
    var myCanvasLeft = document.getElementsByTagName("canvas")[0];
    // 右側のCanvas要素を読み出し
    var myCanvasRight = document.getElementsByTagName("canvas")[1];
    // コンテキストを取得
    var context = myCanvasRight.getContext("2d");
    if (!context){ return; }
    // 映像を小さくして描画する
    context.drawImage(myCanvasLeft, 60, 40, 200, 150);
  }, false);
  // ランダムに左側のCanvasに四角形を描画
  setInterval(function(){
    // 左側のCanvas要素を読み出し
    var myCanvasLeft = document.getElementsByTagName("canvas")[0];
    // コンテキストを取得
    var context = myCanvasLeft.getContext("2d");
```

```
      // 枠の色を赤色に設定
      context.strokeStyle = "red";
      context.lineWidth = 2;
      var x = Math.random() * myCanvasLeft.width;
      var y = Math.random() * myCanvasLeft.height;
      context.strokeRect(x, y, 40, 20);
    }, 100);
  }, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、左側のCanvasの内容が縮小されて右側のCanvasに描画されます。





ONEPOINT 他のCanvasの内容を描画するには「drawImage()」メソッドを使う

他のCanvasの内容を描画するには、「drawImage()」メソッドを使います。「drawImage()」メソッドの最初のパラメータに「canvas」要素を指定します。これでCanvasに指定した「canvas」要素で表示されている内容が描画されます。「drawImage()」メソッドの2番目以降のパラメータは画像を描画する場合と同じで、次のようになります。

◉座標を指定して元のサイズで描画

```
drawImage(canvas, x, y)
```

◉座標とサイズを指定して描画

```
drawImage(canvas, x, y, width, height)
```

◉コピー元のCanvas画像の一部をクリッピングした後で指定した座標とサイズで描画

```
drawImage(canvas, sourceX, sourceY, sourceWidth, sourceHeight, destinationX, destinationY,
 destinationWidth, destinationY)
```

SECTION-220

図形を回転させる

ここでは、Canvasに描く図形を回転させる(描画座標系を回転する)方法について解説します。

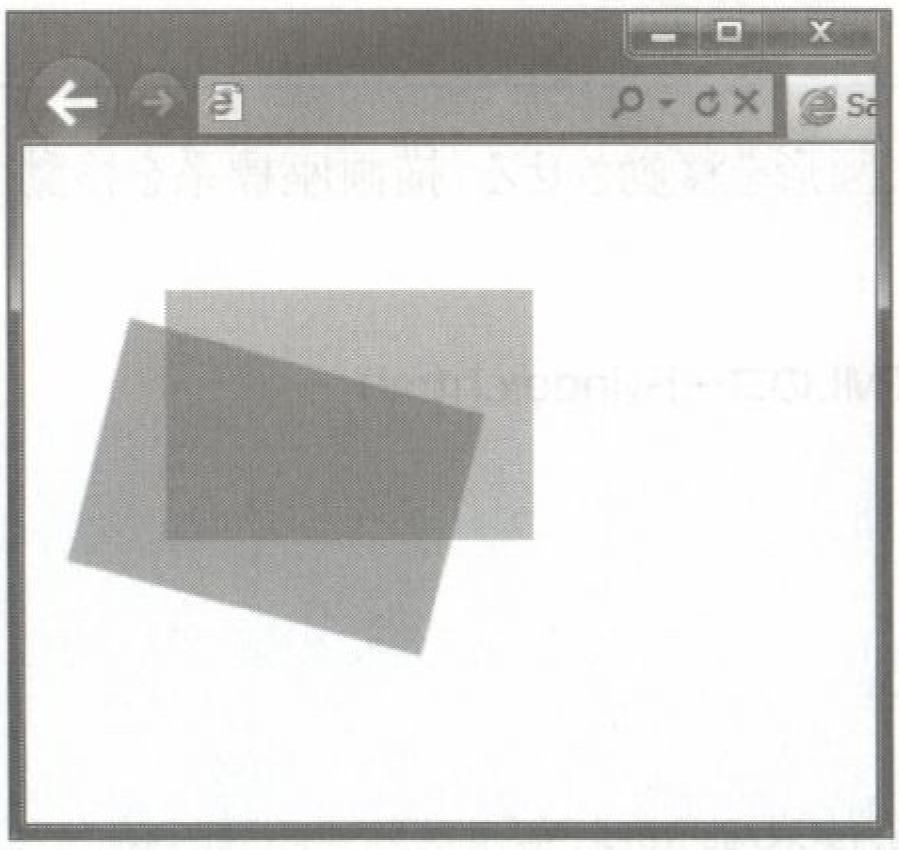
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 塗りの色を設定
  context.fillStyle = "rgba(255, 0, 0, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
  // 15度回転
  context.rotate(15 * Math.PI/180);
  // 塗りの色を設定
  context.fillStyle = "rgba(0, 0, 255, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、回転していない赤色の四角形と、15度回転した青色の四角形が描画されます。



ONEPOINT　回転を行うには「rotate()」メソッドを使う

Canvasの2D Contextで描画する図形を回転させるには、「rotate()」メソッドを使います。パラメータには、回転する角度をラジアンで指定します。「rotate()」メソッドは、原点を基準にして回転します。パラメータが正数の場合は時計回り、負数の場合は反時計回りになります。「rotate()」で回転させると以後、描画する図形はすべて影響を受けます。

「rotate()」メソッドで回転角度指定した後、指定前の状態に戻すには、「rotate()」メソッドの実行前に「save()」メソッドを使って情報を保存しておきます。元に戻す時点で「restore()」メソッドを使います。なお、「restore()」メソッドを実行すると、回転角度以外の情報も元に戻るので注意が必要です。

関連項目 ▶▶▶

- 図形を移動させる……………………………………………………………p.514
- 図形を拡大・縮小させる……………………………………………………p.516
- 図形や画像を変形させる……………………………………………………p.518

SECTION-221

図形を移動させる

ここでは、Canvasに描く図形を移動させる(描画座標系を移動する)方法について解説します。

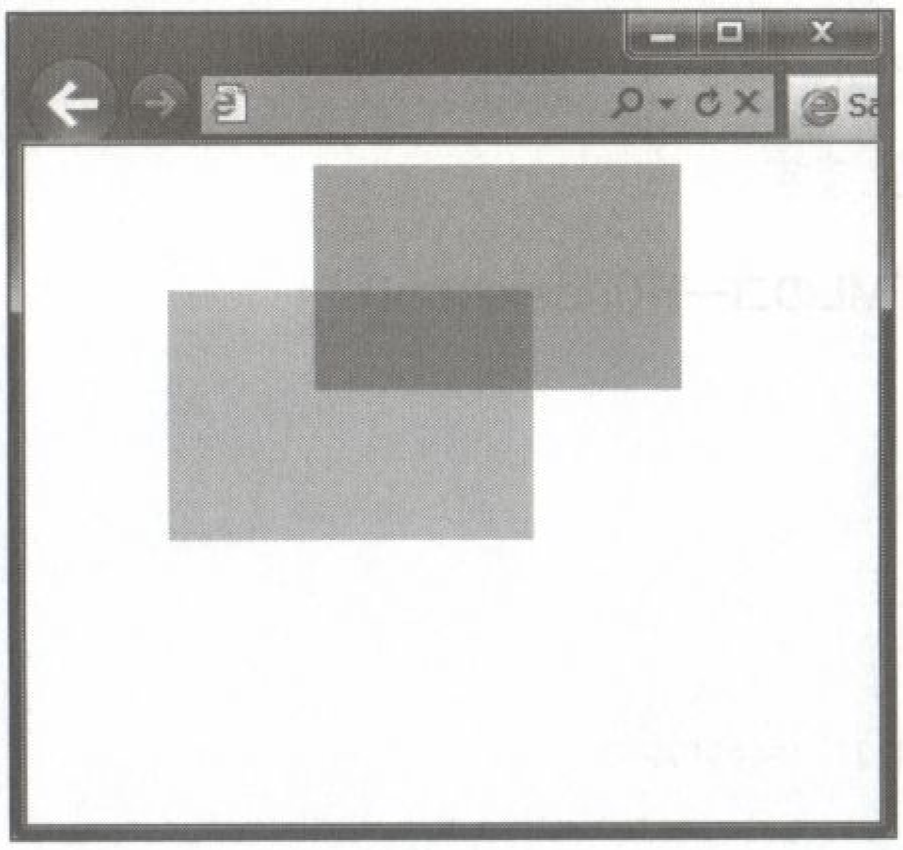
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 塗りの色を設定
  context.fillStyle = "rgba(255, 0, 0, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
  // (60, -60)に移動
  context.translate(60, -60);
  // 塗りの色を設定
  context.fillStyle = "rgba(0, 0, 255, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、「translate()」メソッドで原点を移動させた青色の場所に四角形が描画されます。「translate()」メソッドを使用しない場合には、赤色の場所に描画されます。



ONEPOINT　移動処理を行うには「translate()」メソッドを使う

Canvasの2D Contextで原点を移動させるには、「translate()」メソッドを使います。パラメータには、移動する横のオフセットと縦のオフセットを指定します。パラメータが正数の場合は、値が大きくなるに従って右下に移動します。負数の場合は、値が小さくなるに従って左上に移動します。

「translate()」メソッドで原点を移動した後、指定前の状態に戻すには、「translate()」メソッドの実行前に「save()」メソッドを使って情報を保存しておきます。元に戻す時点で「restore()」メソッドを使います。なお、「restore()」メソッドを実行すると、移動に関する以外の情報も元に戻るので注意が必要です。

関連項目 ▶▶▶

- 図形を回転させる …… p.512
- 図形を拡大・縮小させる …… p.516
- 図形や画像を変形させる …… p.518

SECTION-222

図形を拡大・縮小させる

ここでは、Canvasに描く図形を拡大・縮小（スケーリング）させる（描画座標系をスケーリングする）方法について解説します。

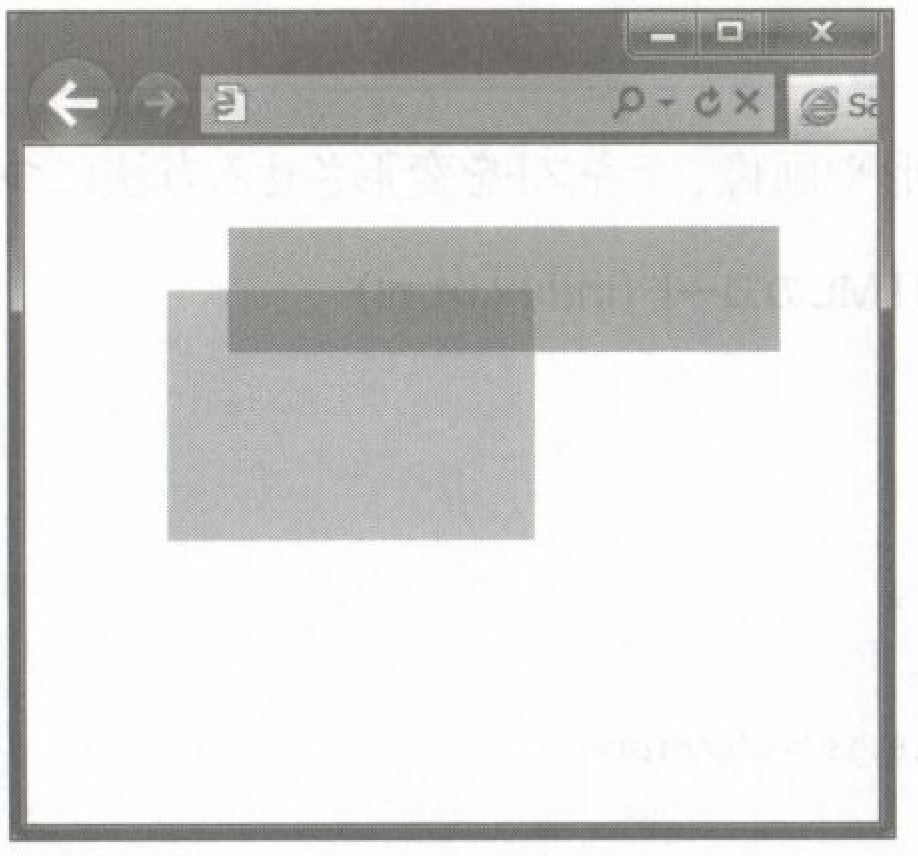
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 塗りの色を設定
  context.fillStyle = "rgba(255, 0, 0, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
  // 横を150%、縦を50%に
  context.scale(1.5, 0.5);
  // 塗りの色を設定
  context.fillStyle = "rgba(0, 0, 255, 0.25)";
  // 四角形を描画
  context.fillRect(50, 50, 150, 100);
}, false);
```

ブラウザでindex.htmlを表示すると、拡大・縮小した図形が青色の場所に描画されます。拡大・縮小していない図形は赤色の場所に描画されます。



ONEPOINT **拡大・縮小を行うには「scale()」メソッドを使う**

Canvasの2D Contextで図形を拡大・縮小させるには、「scale()」メソッドを使います。パラメータには、倍率を指定します。倍率は数値で指定し、「1.0」が100%サイズ、「0.5」なら50%、「2.0」なら200%を示します。

「scale()」メソッドで拡大・縮小すると、原点を基準にして行われます。原点は、図形の左上や中心の座標ではない点には注意してください。

また、「scale()」メソッドで拡大・縮小した後、指定前の状態に戻すには、「scale()」メソッドの実行前に「save()」メソッドを使って情報を保存しておきます。元に戻す場合には、「restore()」メソッドを使います。なお、「restore()」メソッドを実行すると拡大・縮小に関する以外の情報も元に戻るので注意が必要です。

関連項目 ▶▶▶

● 図形を回転させる……p.512
● 図形を移動させる……p.514
● 図形や画像を変形させる……p.518

SECTION-223

図形や画像を変形させる

ここでは、描画する図形や画像、テキストを変形させる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

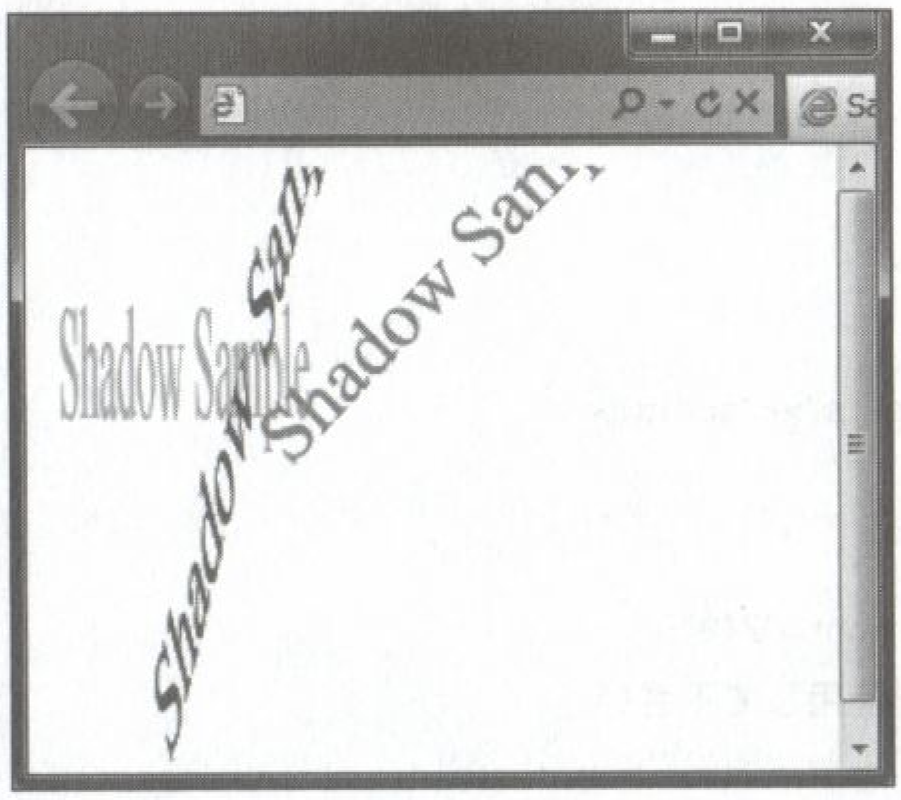
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // フォントとサイズを設定する
  context.font = "24pt 'Tahoma-Bold'";
  // 変形する
  context.transform(0.5, 0, 0, 2, 0, 0);
  // 文字を描画する
  context.fillStyle = "orange";
  context.fillText("Shadow Sample", 10, 50);
  // 現在の変形に加えてさらに変形する
  var rad = -45 * Math.PI / 180;
  var ks = Math.sin(rad);
  var kc = Math.cos(rad);
  context.transform(kc, ks, -ks, kc, 100, 120);
  // 文字を描画する
  context.fillStyle = "blue";
  context.fillText("Shadow Sample", 0, 0);
  // 現在の変形を破棄して新たにマトリックスを設定
  context.setTransform(kc, ks, -ks, kc, 100, 120);
  // 文字を描画する
  context.fillStyle = "red";
```

```
    context.fillText("Shadow Sample", 0, 0);
}, false);
```

ブラウザでindex.htmlを表示すると、変形設定した文字が描画されます。ここでは、スケールを指定した文字、さらに回転を加えた文字、新たに回転のみを指定して文字を描画しています。



ONEPOINT

図形や画像を変形させるには「transform()」メソッドや「setTransform()」メソッドを使う

描画する図形や画像は変形させるメソッドには、「transform()」メソッドと「setTransform()」メソッドがあります。それぞれ、書式は、次のようになります。

```
transform(m11, m12, m21, m22, dx, dy)

setTransform(m11, m12, m21, m22, dx, dy)
```

「transform()」メソッドは、現在の変形に、さらに指定した変形を加えます。「setTransform()」は、現在の変形設定を初期化し、新たに指定したマトリクスで変形を設定します。変形マトリクスは、次のようになります。

m11	m21	dx
m12	m22	dy
0	0	1

「transform()」メソッドは、「scale()」メソッド、「rotate()」メソッド、「translate()」メソッドでの変形を一括して行うことができます。「setTransform()」は、これらの変形設定を初期化する場合にも使うことができます。

関連項目 ▶▶▶

- 図形を回転させる……p.512
- 図形を移動させる……p.514
- 図形を拡大・縮小させる……p.516

SECTION-224

パス内に座標点があるか調べる

ここでは、パス内に座標点があるか調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

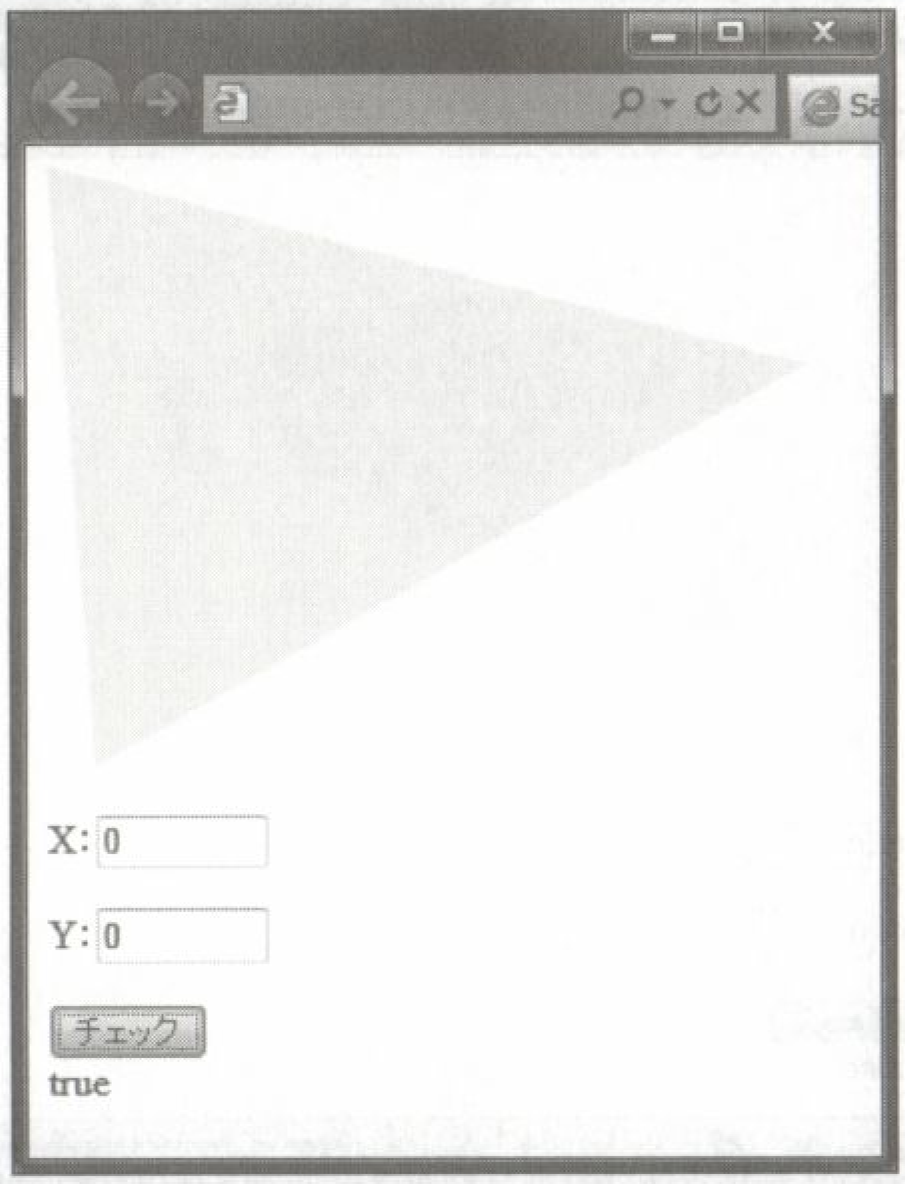
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
    <form>
      <p>X:<input type="text" size="6" value="0" id="xp"></p>
      <p>Y:<input type="text" size="6" value="0" id="yp"></p>
      <input type="button" value="チェック" id="checkBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

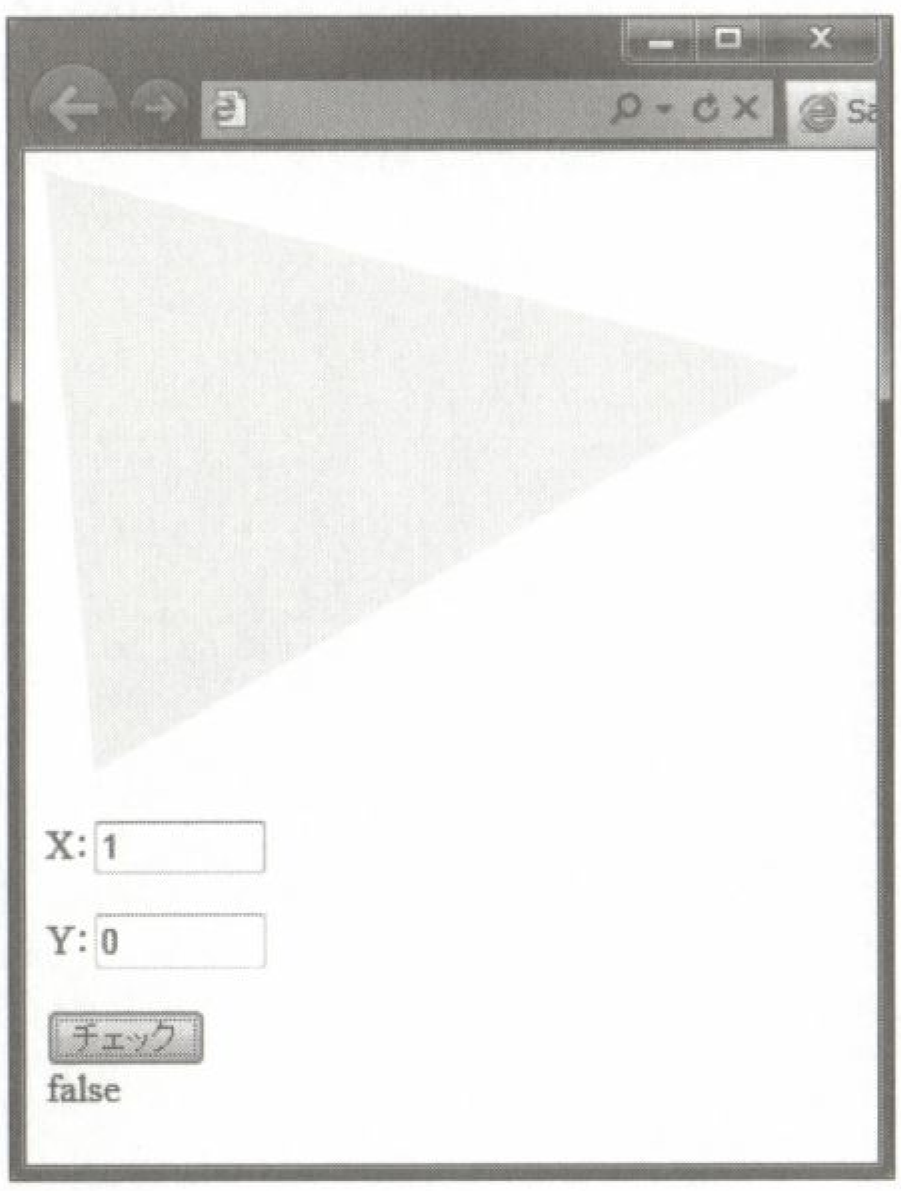
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 新規にパスを作成し、三角形を描く
  context.beginPath();
  context.moveTo(0,0);
  context.lineTo(310, 80);
  context.lineTo(20, 240);
  context.closePath();
  context.fillStyle = "rgba(255,255,0, 0.5)";
  context.fill();
  // ボタンがクリックされたら指定した座標がパス内かどうか調べる
  document.getElementById("checkBtn").addEventListener("click", function(){
    // 入力されたX, Y座標値を読み出し
    var x = document.getElementById("xp").value;
```

```
    var y = document.getElementById("yp").value;
    // X, Y座標が現在のパス内かどうか調べる
    var flag = context.isPointInPath(x, y);
    // 結果を出力
    ele.innerHTML = flag;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに座標値を入力してボタンをクリックすると、指定した座標値がパス内かどうかを返します。パス内の場合は、次のように「true」になります。



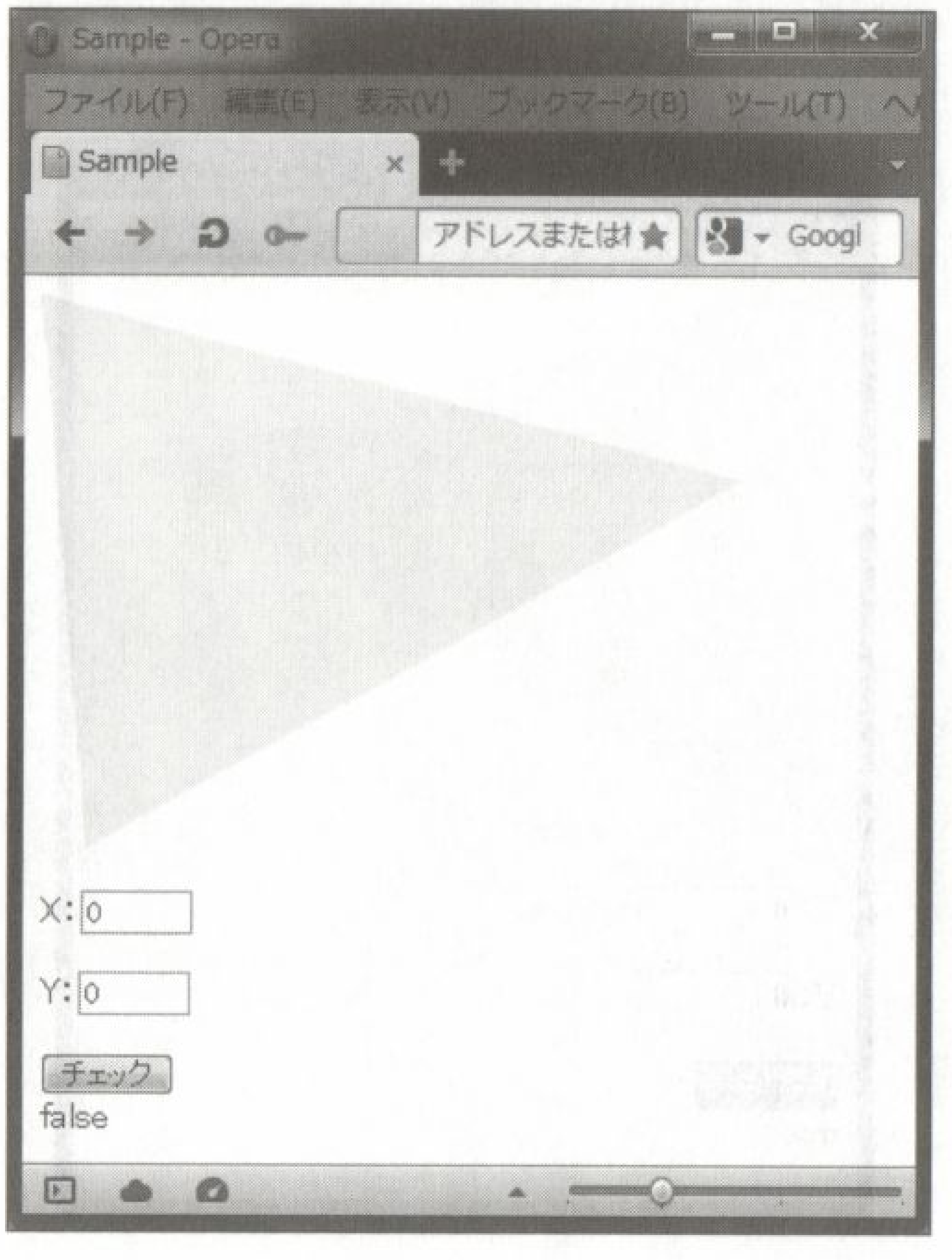
パス外の場合は、「false」になります。



ONEPOINT パス内に座標点があるか調べるには「isPointInPath()」メソッドを使う

パス内に座標点があるか調べるには、「isPointInPath()」メソッドを使います。パラメータには、調べたいX座標とY座標を指定します。パス内であれば「true」、パス外であれば「false」を返します。調べるパスは、「beginPath()」メソッドを使わない限り、最後に作成したパスが対象になります。

なお、Opera 11.0〜12.0では、パスの開始点など、本来パス内であるにもかかわらず、「false」となってしまう場合もあります（下図参照）。



SECTION-225

Canvasの状態を保存・復元する

ここでは、Canvasの状態を保存・復元する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

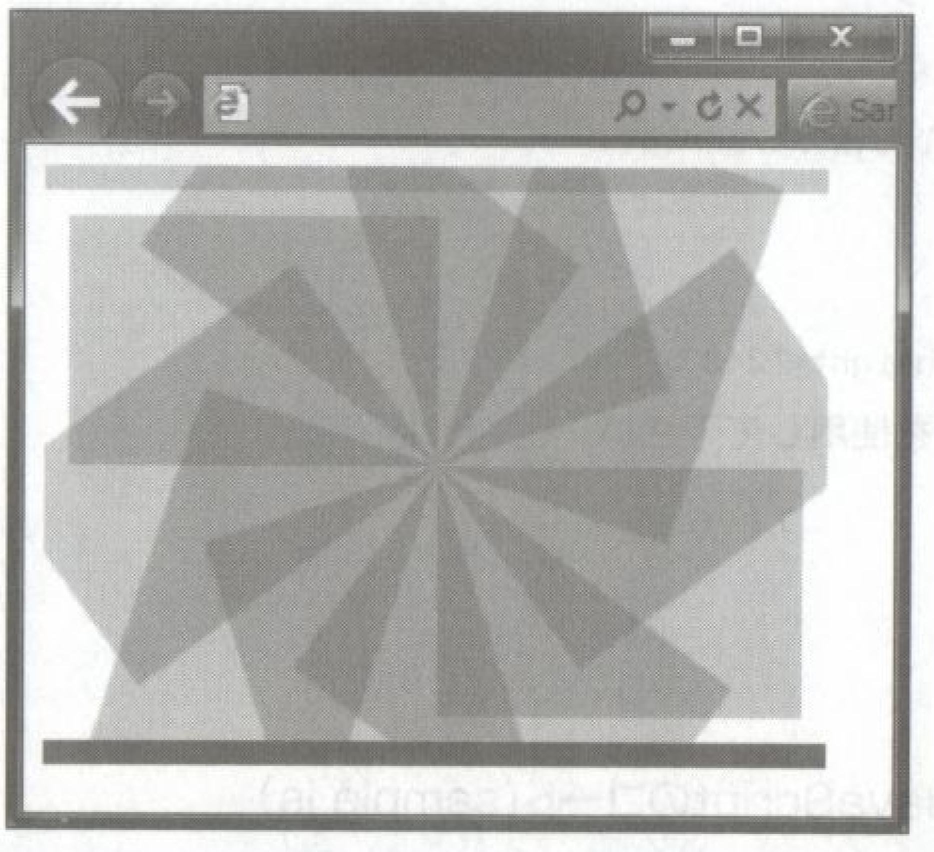
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 塗りの色を設定
  context.fillStyle = "green";
  // 状態を保存
  context.save();
  // 不透明度を設定
  context.globalAlpha = 0.2;
  // 四角形を回転描画(10回繰り返す)
  for(var i=0; i<360; i+=36){
    // 状態を保存
    context.save();
    // 移動&回転
    context.translate(160, 120);
    context.rotate(i * Math.PI/180);
    // 四角形を描画
    context.fillRect(0, 0, 150, 100);
    // 状態を復元
    context.restore();
  }
```

```
    // 最上部に四角形を描画
    context.fillRect(0, 0, 320, 10);
    // 状態を復元
    context.restore();
    // 最下部に四角形を描画
    context.fillRect(0, 230, 320, 10);
}, false);
```

ブラウザでindex.htmlを表示すると、「save()」メソッドと「restore()」メソッドを使い、図形がCanvasの中心で回転するように描画されます。



ONEPOINT Canvasの状態を保存するには「save()」メソッド、復元するには「restore()」メソッドを使う

Canvasでは、「save()」メソッドを使うことで、現在のCanvasの状態を保存することができます。「save()」により保存した情報を元に戻すには、「restore()」メソッドを使います。「save()」メソッドを実行した場合、Canvas内のピクセル情報は保存されません。保存されるのは、次のプロパティになります。

- strokeStyle
- fillStyle
- globalAlpha
- lineWidth
- lineCap
- lineJoin
- miterLimit
- shadowOffsetX
- shadowOffsetY
- shadowBlur
- shadowColor
- globalCompositeOperation
- font
- textAlign
- textBaseline

SECTION-226

Canvasに描画済みの内容をスクロールする

ここでは、Canvasに描画済みの内容をスクロールする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

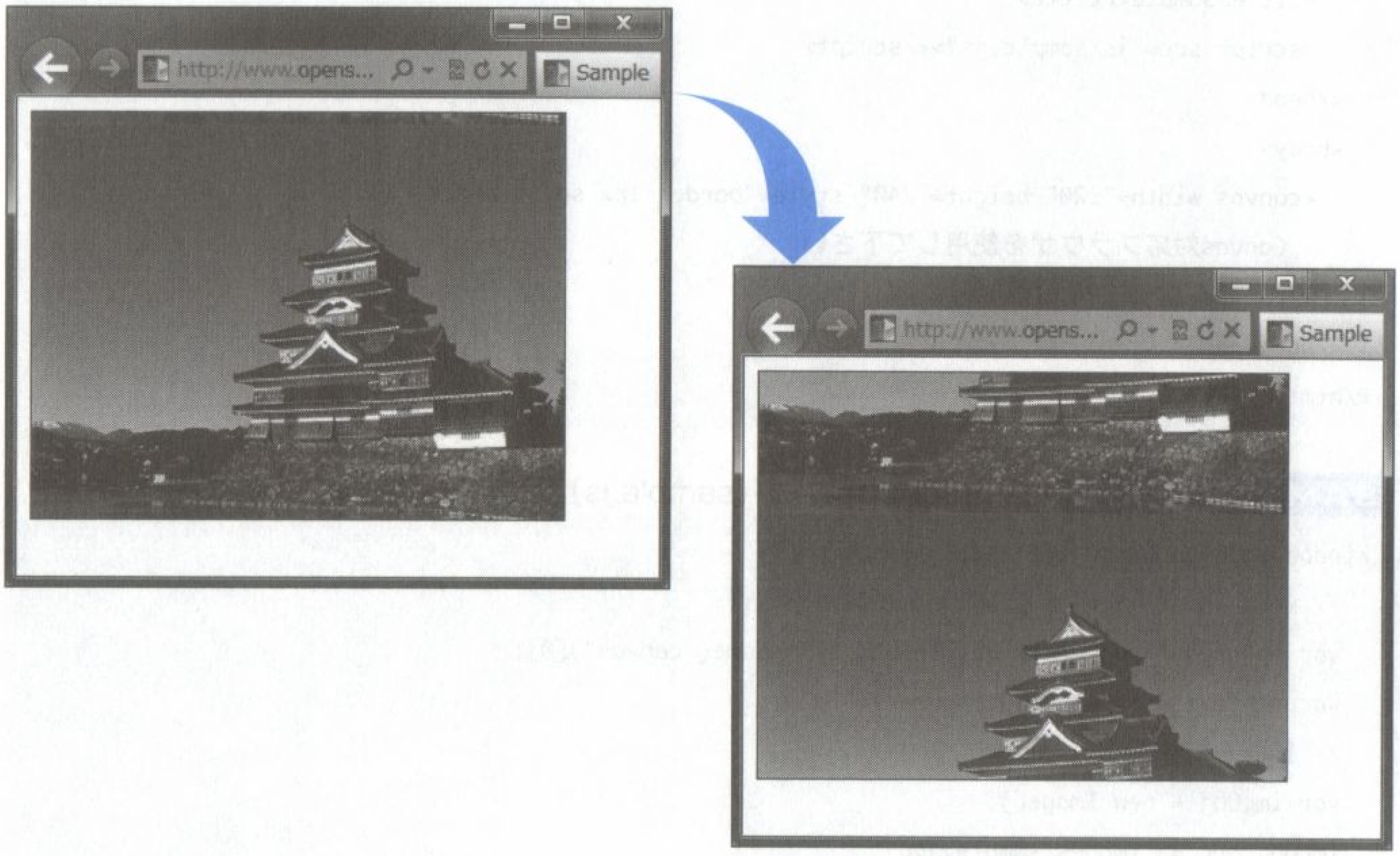
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240" style="border:1px solid black"">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // 描画を描画する
  var imgObj = new Image();
  imgObj.src = "images/sample.jpg";
  imgObj.onload = function(){
    // 画像が読み込まれたらCanvasに描画
    context.drawImage(this, 0, 0);
    // スクロール処理を行う
    setInterval("scrollDown()", 100);
  }
}, false);
// スクロール処理(上から下)
function scrollDown(){
  // Canvas要素を読み出し2Dコンテキストを取得
  var myCanvas = document.getElementsByTagName("canvas")[0];
  var context = myCanvas.getContext("2d");
  // Canvasの幅を取得
  var w = myCanvas.width;
  var h = myCanvas.height;
  // Canvas内のピクセルデータを読み出す
  var pixelData = context.getImageData(0,0, w, h);
```

```
    // 一番下のピクセルデータを読み出す
    var pixelDataLast = context.getImageData(0,h-1, w, 1);
    // それぞれを特定の座標に描画する
    // 大きな画像はY=1の座標に描画
    context.putImageData(pixelData, 0,1);
    // 一番下のピクセルはY=0の座標に描画
    context.putImageData(pixelDataLast, 0,0);
}
```

ブラウザでindex.htmlを表示し、ページが読み込まれると、Canvasに描画された画像が上から下にスクロールします。



ONEPOINT Canvasの内容をスクロールするには「getImageData()」メソッドと「putImageData()」メソッドを使う

Canvasに描画済みの画像や図形をスクロールするには、2Dコンテキストの「getImageData()」メソッドと「putImageData()」メソッドを利用します。まず、「getImageData()」メソッドを使って、Canvas内の画像データを取得します。このとき、スクロールして消える端のピクセル部分は別に取得しておきます。

次に、「putImageData()」メソッドを使って、座標をずらして描画します。これでCanvas画像がスクロールしたことになります。ただし、このままでは端のピクセル情報が消されてしまいます。そこで、先ほど、「getImageData()」メソッドで取得しておいた端のピクセル部分を描画します。これで画像がスクロールしたことになります。タイマーを使ってスクロール処理を呼び出せば、Canvasに描画された画像が延々とスクロールすることになります。

SECTION-227

Canvasの画像データをdataURL形式に変換する

ここでは、Canvasの画像データをdataURL形式に変換する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 画像を描画する
  var imgObj = new Image();
  imgObj.src = "images/matsumoto.jpg";
  imgObj.onload = function(){
    context.drawImage(this, 0, 0);
    // ピクセルデータを読み出しPNG形式にする
    var data = myCanvas.toDataURL("image/png");
    // 変換結果を画面に出力
    ele.innerText = ele.textContent = data;
  }
}, false);
```

ブラウザでindex.htmlを表示すると、Canvas内のピクセルデータが変換された結果が表示されます。



ONEPOINT ピクセルデータをdatURL形式に変換するには「toDataURL()」メソッドを使う

Canvasに描画された内容をdataURL形式に変換するには、「toDataURL()」メソッドを使います。「toDataURL()」メソッドのパラメータを省略した場合は、PNG形式の画像に変換されます。「image/jpeg」とすると、JPEG形式に変換されます。ただし、PNG形式以外では、ブラウザによって変換されないことがあります。

「toDataURL()」メソッドによって変換されたデータは、「img」要素の「src」プロパティなどに代入して画像として表示することができます。これを利用して、Canvasでお絵かきソフトを作成した場合の取り消し機能や、File APIを使ってデータを保存することができます。

COLUMN ローカルディスク上での動作

Google Chromeなどのブラウザではセキュリティの関係で、ローカルディスク上で動作している場合、画像データを描画できても読み出すことができません。このため、「toDataURL()」メソッドを実行しても画像データとして変換することはできません。

関連項目 ▶▶▶

● 画像をローカルストレージに保存する ………………………… p.529

SECTION-228

画像をローカルストレージに保存する

ここでは、画像をローカルストレージに保存する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" id="save" value="ローカルストレージに保存">
    </form>
    <canvas width="320" height="240">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

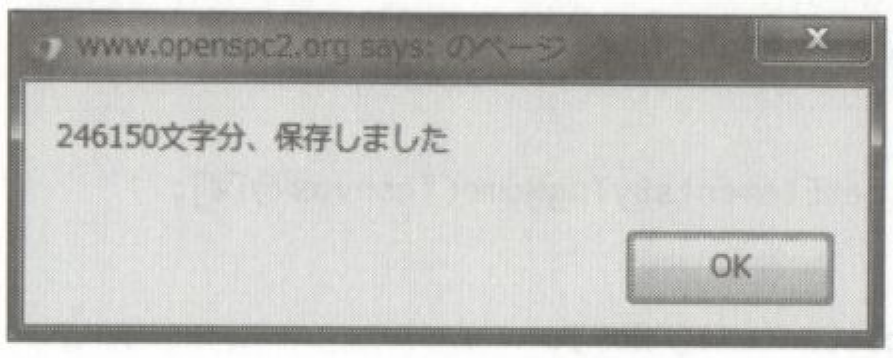
```
window.addEventListener("load", function(){
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // コンテキストを取得
  var context = myCanvas.getContext("2d");
  if (!context){ return; }
  // 画像を描画する
  var imgObj = new Image();
  imgObj.src = "images/sunflower.jpg?1";
  imgObj.onload = function(){
    // 座標を指定して元のサイズで描画
    context.drawImage(this, 0, 0);
    // 画像のまわりに白い半透明の枠を描画
    context.strokeStyle = "rgba(255,255,255, 0.7)";
    context.lineWidth = 10;
    context.strokeRect(5,5, 310, 230);
  }
  // ボタンにイベントを設定
  document.getElementById("save").addEventListener("click", function(){
    // ピクセルデータを文字列に変換
    var data = myCanvas.toDataURL("image/png");
```

```
      // ローカルストレージに保存
      localStorage.setItem("pixelData", data);
      // 確認のダイアログを表示
      alert(data.length+"文字分、保存しました");
    }, false);
  }, false);
```

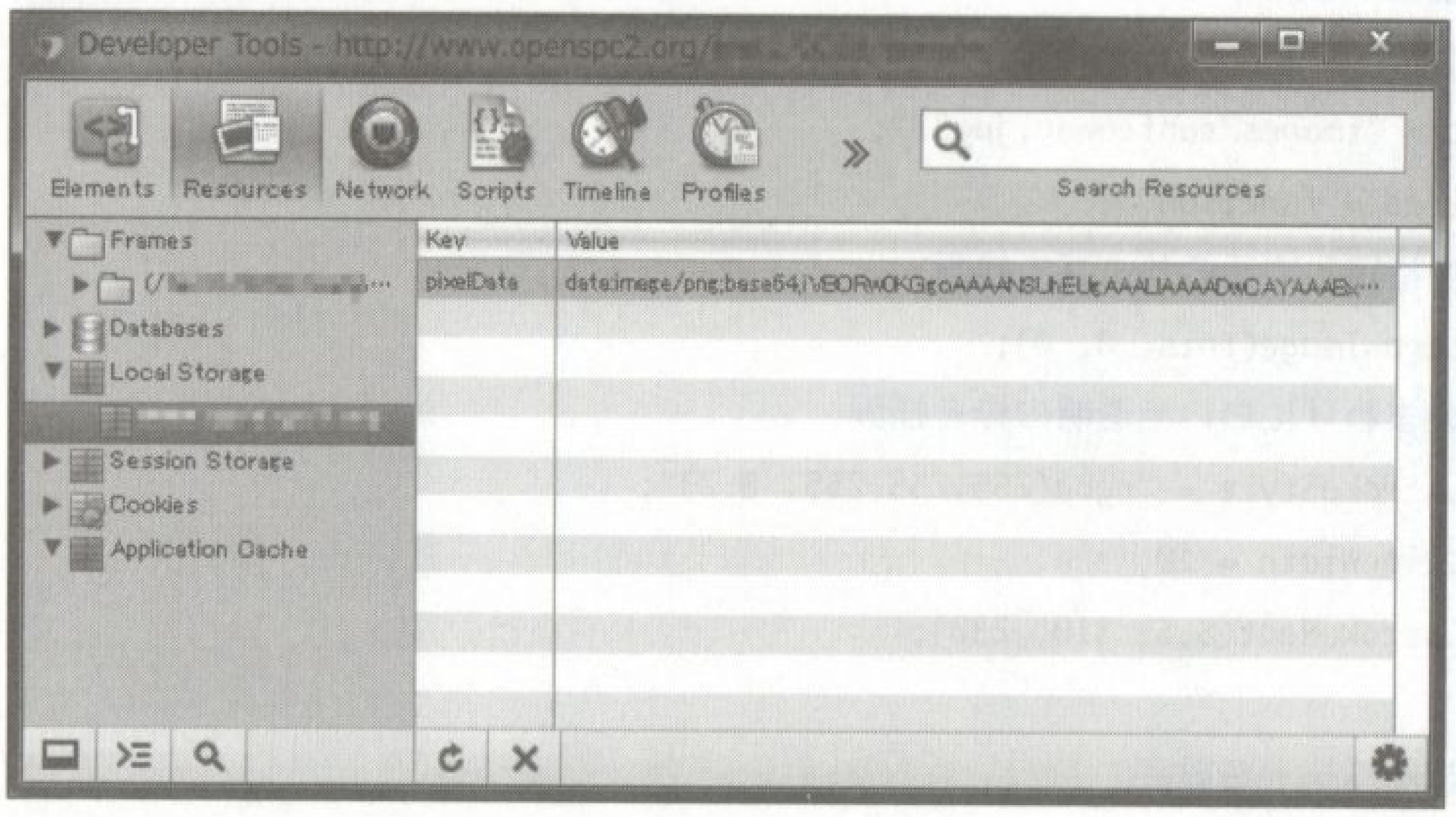
ブラウザでindex.htmlを表示するとCanvasに画像が表示され、その後、ボタンをクリックするとローカルストレージに画像データが保存されます。



保存されると、保存した文字数がアラートダイアログに表示されます。



保存内容は、デバッガを使って確認することができます。



ONEPOINT 画像を保存するには「toDataURL()」メソッドを使う

Canvasに表示されている画像データをローカルストレージに保存するには、「toDataURL()」メソッドを利用します。これは、Canvasに用意されているメソッドで、画像データを文字列(BASE64形式)に変換します。「toDataURL()」メソッドのパラメータには、変換する画像形式のMIMEタイプを指定します。省略すると、「image/png」になります。また、「image/jpeg」とするとJPEG形式のデータになりますが、ブラウザによっては対応していない場合もあります。

文字列に変換してしまえば、後は「localStorage」の「setItem()」メソッドを使ってキー名を付けてデータを保存します。保存したデータを読み出し、再度、Canvasに描画するには「Image」オブジェクトを作成し、その「src」プロパティにローカルストレージに保存したCanvasデータを設定します。その後、画像が読み込まれたら「drawImage()」メソッドを使い、Canvas上に描画します。

COLUMN ローカルディスク上での動作

Google Chromeなどのブラウザではセキュリティの関係で、ローカルディスク上で動作している場合、画像データを描画できても読み出すことができません。このため、サンプルのように、「toDataURL()」メソッドを実行しても画像データとして変換することはできません。

関連項目 ▶▶▶

● Canvasの画像データをdataURL形式に変換する ………… p.527

SECTION-229

WebGLが利用できるか調べる

ここでは、Canvasで3D描画を行う際に必要なWebGLが使用可能か調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>WebGLチェック</h3>
    <div><output></output></div>
    <canvas width="480" height="64">
      Canvas対応ブラウザを使用して下さい
    </canvas>
  </body>
</html>
```

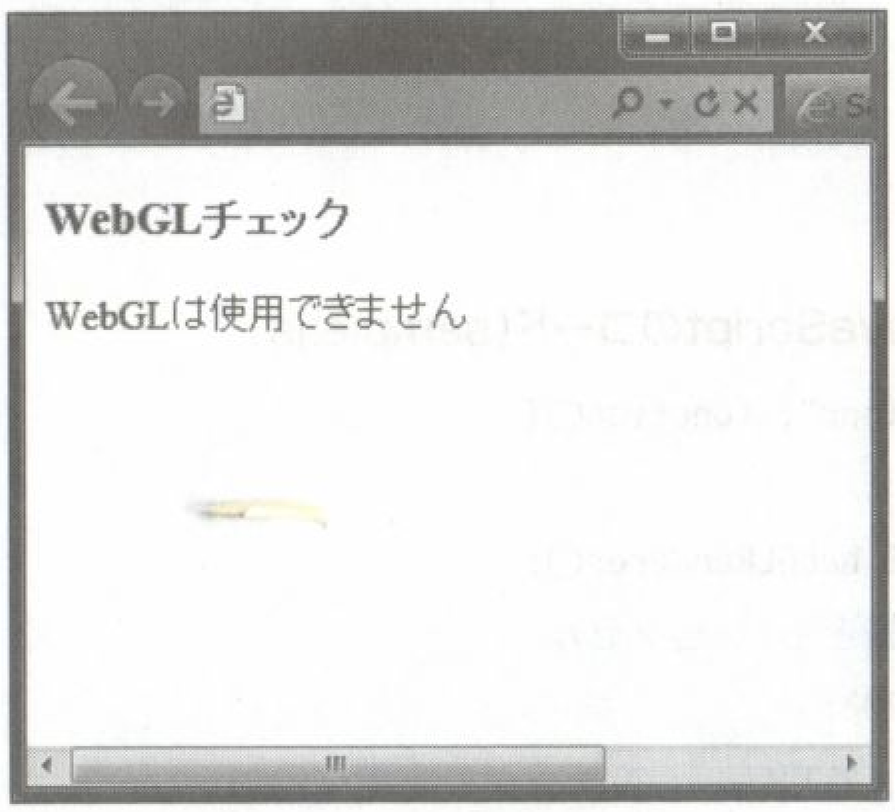
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Canvas要素を読み出し
  var myCanvas = document.getElementsByTagName("canvas")[0];
  // WebGLコンテキストを取得
  var context = myCanvas.getContext("webgl") || myCanvas.getContext("experimental-webgl");
  if (context == null){
    ele.innerHTML = "WebGLは使用できません";
  }else{
    ele.innerHTML = "WebGLは使用できます";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、WebGLが使用可能かどうか調べた結果が表示されます。Google ChromeではWebGLが利用できるので、使用可能であるとの文字が表示されます。



IE9ではWebGLが利用できないので、使用できないと表示されます。



ONEPOINT　3D描画が可能か調べるには「getContext()」メソッドを使う

Canvasで3D描画が可能かどうか調べるには、「getContext()」メソッドを使います。Canvasで3D描画を行う際にはWebGLが利用されるので、「getContext()」メソッドのパラメータに「"webgl"」、または、「"experimental-webgl"」の文字列を指定します。

なお、1つのCanvasには2Dか3Dのどちらか一方しか利用することはできませんが、2Dと3DのCanvasを重ね合わせて表示することはできます。

「getContext()」メソッドで指定している「"experimental-webgl"」は、仕様が確定するまでに使用される一時的なものです。仕様が確定した場合は、「"webgl"」となります。

関連項目 ▶▶▶

● WebGLが使えないデバイスでも3D描画を可能にする……………………………p.589

Three.jsを使って3D処理を行う

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って3D処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

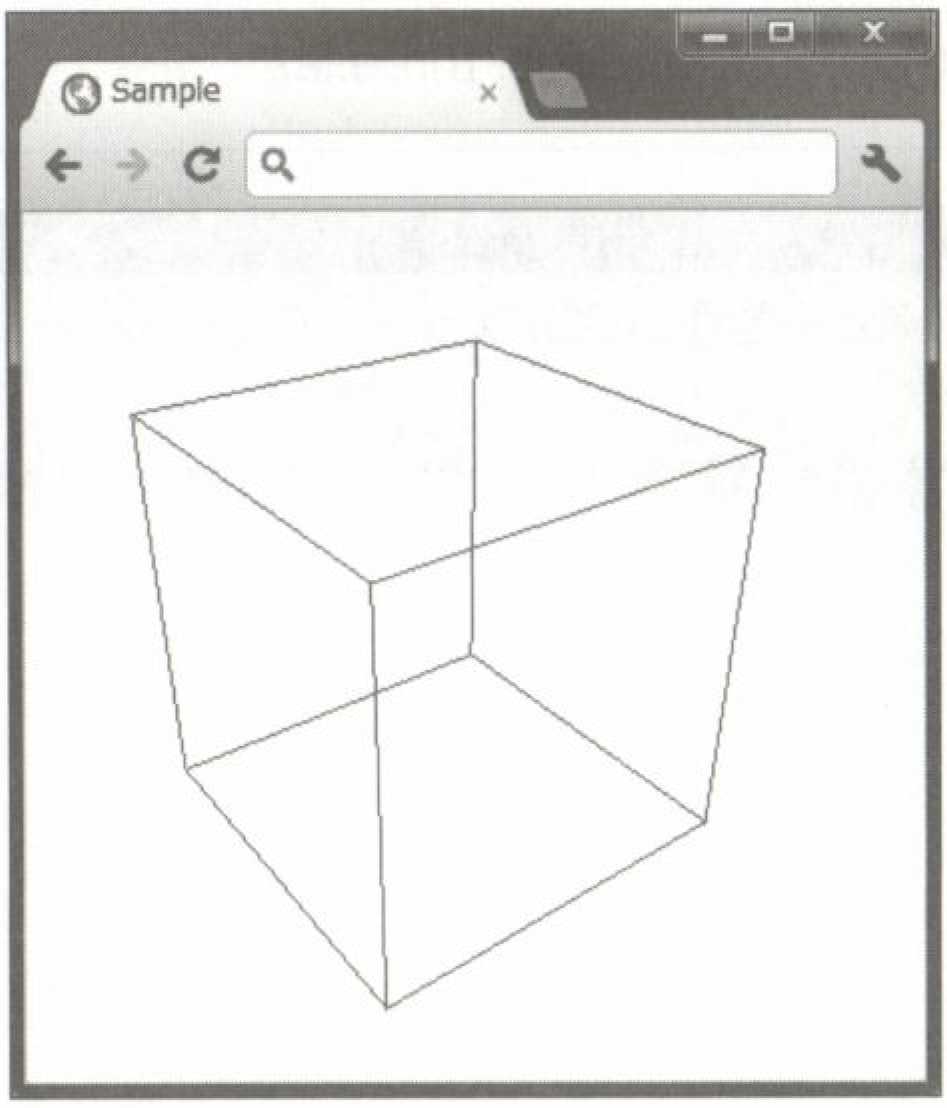
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(55, 40, 40);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 四角形のXYZサイズを設定
    new THREE.CubeGeometry(30, 30, 30),
    // ワイヤーフレーム表示。色は青
    new THREE.MeshBasicMaterial({
      "color" : 0x0000ff,
      "wireframe" : true
        })
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
```

```
    // カメラを3Dシーンに追加
    scene.add(camera);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、青色でワイヤーフレームの立方体が描画されます。



ONEPOINT 3D処理を簡単に行うにはThree.jsを使う

CanvasでのWebGLを利用して3D処理を行うには、かなり多くの手順が必要になります。面倒な3D処理を簡単に行えるライブラリとして、Three.jsがあります。Three.jsを利用して3D処理を行うには、次のようにします。

まず、Three.jsファイルを読み込みます。他に必要なファイルはありません。

「new THREE.WebGLRenderer()」としてレンダリングするためのオブジェクトを生成します。レンダリング領域のサイズは「setSize()」メソッドで行い、横幅と縦幅をピクセル数で指定します。生成されたオブジェクトにはCanvas要素が含まれているので、これをページ上に出力します。また、デフォルトではアンチエイリアスがかからないので、必要であれば「THREE.WebGLRenderer()」のパラメータに「{"antialias" : true}」を指定します。

次にカメラの設定を行います。Three.jsには、平行法と遠近法を利用したカメラが用意されています。3Dで多く見られるシーンは、遠近法を利用したものになっています。Three.jsでは、「new THREE.PerspectiveCamera()」として遠近法を利用したカメラを生成できます。カメラ位置は「position.set()」メソッドを使い、X座標とY座標、Z座標を指定します。また、「lookAt()」メソッドを使って計算処理を行います。

次に描画する立方体を生成します。立方体の生成は、「new THREE.Mesh()」として行い、パラメータには生成する立方体のサイズとマテリアルを指定します。

次にシーンを「new THREE.Scene()」として生成します。生成したシーンに、先ほど

作成した立方体とカメラを「add()」メソッドを使って追加します。

後は「new THREE.WebGLRenderer()」として生成したオブジェクトの「render()」メソッドを使って、実際の描画・表示を行います。「render()」メソッドのパラメータには、シーンオブジェクトとカメラオブジェクトを指定します。

なお、Three.jsの入手先は、次のようになります。

● Three.js

URL https://github.com/mrdoob/three.js/

関連項目 ▶▶▶

● ワイヤーフレーム表示の幅と不透明度を設定する ……… p.537
● カメラを設定する ……… p.539
● 光源を追加する ……… p.542

SECTION-231

ワイヤーフレーム表示の幅と不透明度を設定する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使ってワイヤーフレーム表示の幅と不透明度を設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

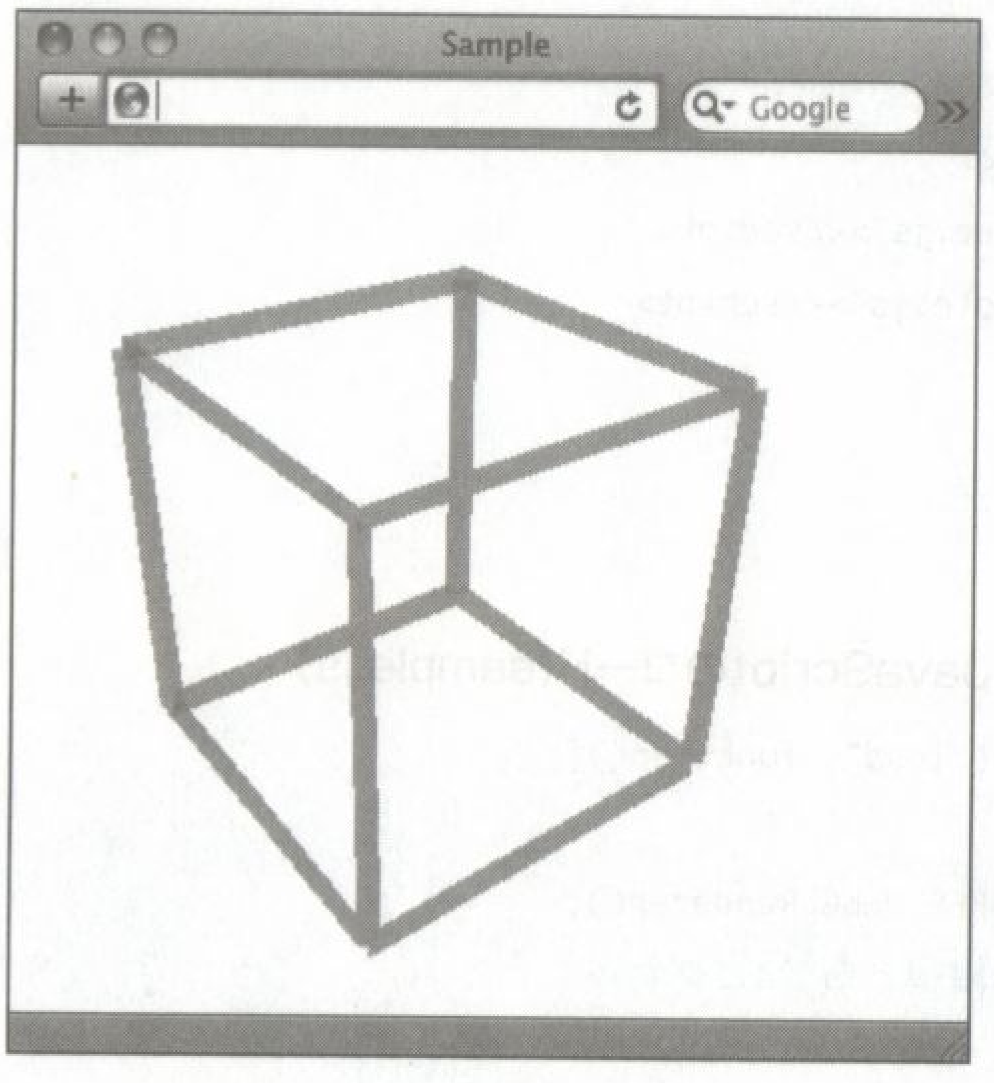
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(55, 40, 40);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 四角形のXYZサイズを設定
    new THREE.CubeGeometry(30, 30, 30),
    // ワイヤーフレーム表示。色は赤。不透明度は0.25%
    new THREE.MeshBasicMaterial({
      color : 0xff0000,
      wireframe : true,
      wireframeLinewidth : 10,
      opacity: 0.25
          })
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
```

```
    // 立方体を3Dシーンに追加
    scene.add(cube);
    // カメラを3Dシーンに追加
    scene.add(camera);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、赤色でワイヤーフレームの立方体が描画されます。枠の幅は10に設定しています。



ONEPOINT ワイヤーフレームの幅を指定するには「wireframeLinewidth」プロパティ、不透明度を指定するには「opacity」プロパティを使う

Three.jsで表示する物体をワイヤーフレームにするには、マテリアルを設定するオブジェクトで「wireframe」プロパティに「true」を設定します。ワイヤーフレームの線の幅を指定するには、「wireframeLinewidth」プロパティに単位なしの値を指定します。

表示する物体全体の不透明度を指定するには、「opacity」プロパティに「0.0」～「1.0」までの小数値を指定します。「0」だと完全な透明になり、「1.0」だと完全な不透明になります。

なお、Windows 7上では、Google Chrome 21、Firefox 14とも、「wireframeLinewidth」プロパティの値は反映されず、ワイヤーフレームの線幅は「1」になります。

関連項目 ▶▶▶

- Three.jsを使って3D処理を行う ………… p.534
- カメラを設定する ………… p.539
- 光源を追加する ………… p.542

SECTION-232

カメラを設定する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使ってカメラを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

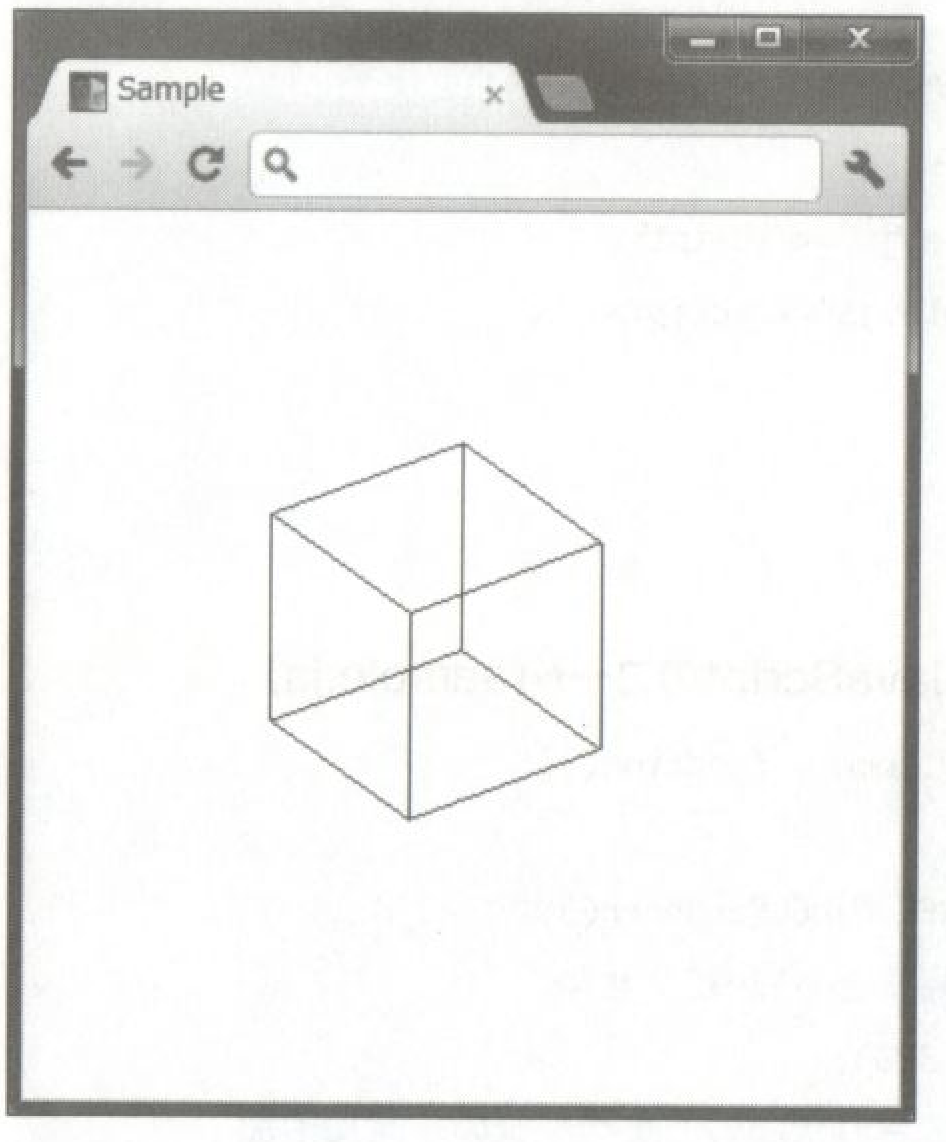
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

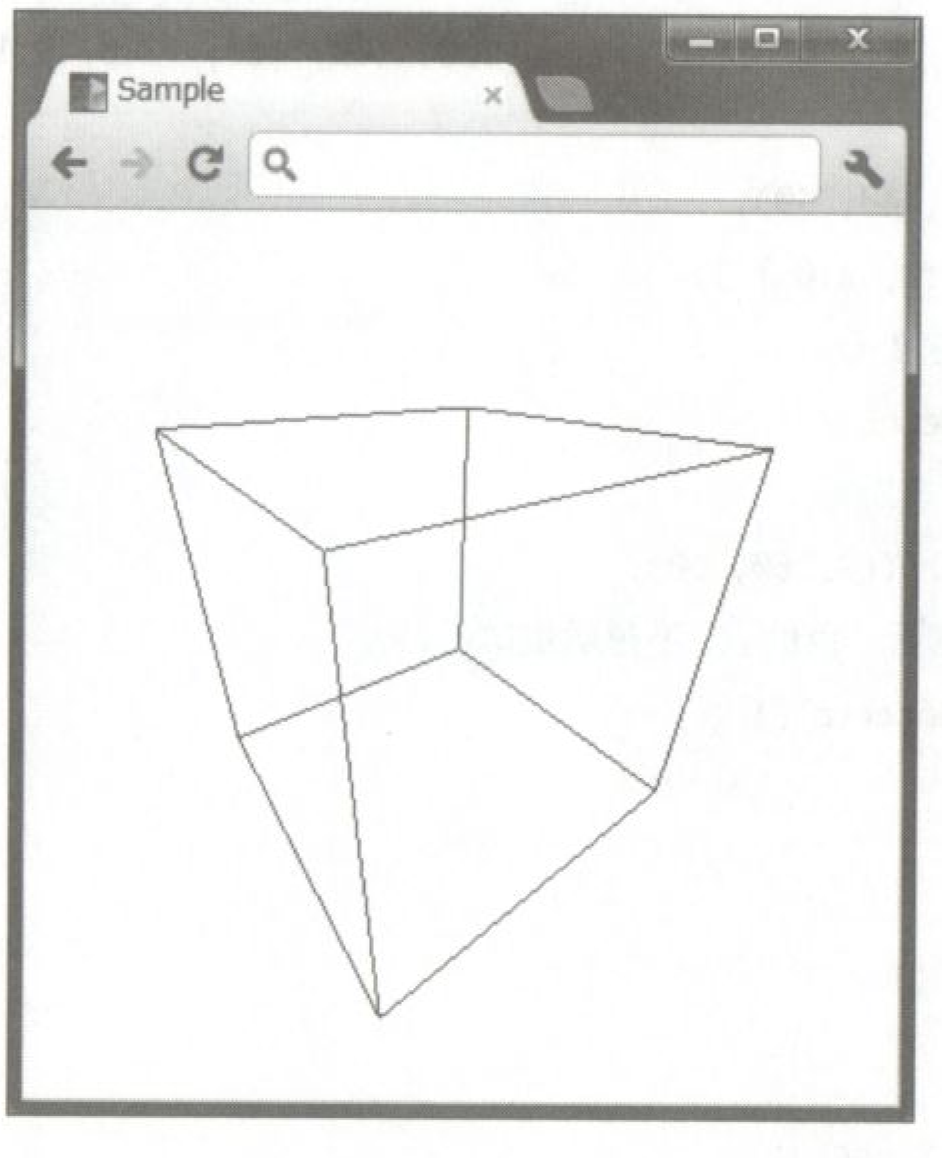
```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  // var camera = new THREE.PerspectiveCamera(80, 1.0, 1, 10000);
  var camera = new THREE.OrthographicCamera(-100, 100, 100, -100, 0.1, 2000);
  // カメラ位置を設定
  camera.position.set(55, 40, 40);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 四角形のXYZサイズを設定
    new THREE.CubeGeometry(60, 60, 60),
    // ワイヤーフレーム表示。色は赤。不透明度は0.25%
    new THREE.MeshBasicMaterial({
      color : 0x000000,
      wireframe : true,
          })
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
```

```
    // 立方体を3Dシーンに追加
    scene.add(cube);
    // カメラを3Dシーンに追加
    scene.add(camera);
    // レンダリング
    renderer.render(scene, camera);
  }, false);
```

ブラウザでindex.htmlを表示すると、正投影のカメラの場合の立方体が表示されます。ここでは、わかりやすいようにワイヤーフレーム表示にしています。



透視投影のカメラの場合の立方体は次のように表示されます。画角にもよりますが、手前に来るほど図形が両側に広がり、歪みます。



ONEPOINT 設定できるカメラには透視投影のカメラと正投影のカメラ、複合カメラがある

映画やアニメなどで見られるCGでは、透視投影のカメラが使われます。透視投影のカメラは「new THREE.PerspectiveCamera()」として生成します。パラメータは4つあり、最初が画角、次が縦横比、次がカメラから手前までの距離、最後がカメラから奥までの距離となっています。

製図やCADなどで使われるカメラは、正投影のカメラが使われます。正投影の場合は透視投影と異なり、図形が距離によって歪むことはありません。正投影のカメラは、「new THREE.OrthographicCamera()」として生成します。パラメータは、表示する範囲の左座標、右座標、上座標、下座標、カメラから手前までの距離、最後がカメラから奥までの距離となっています。

他にも複合カメラとして、「THREE.CombinedCamera()」が用意されています。

関連項目 ▶▶▶

- Three.jsを使って3D処理を行う …… p.534
- ワイヤーフレーム表示の幅と不透明度を設定する …… p.537
- 光源を追加する …… p.542

SECTION-233

光源を追加する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って光源を追加する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

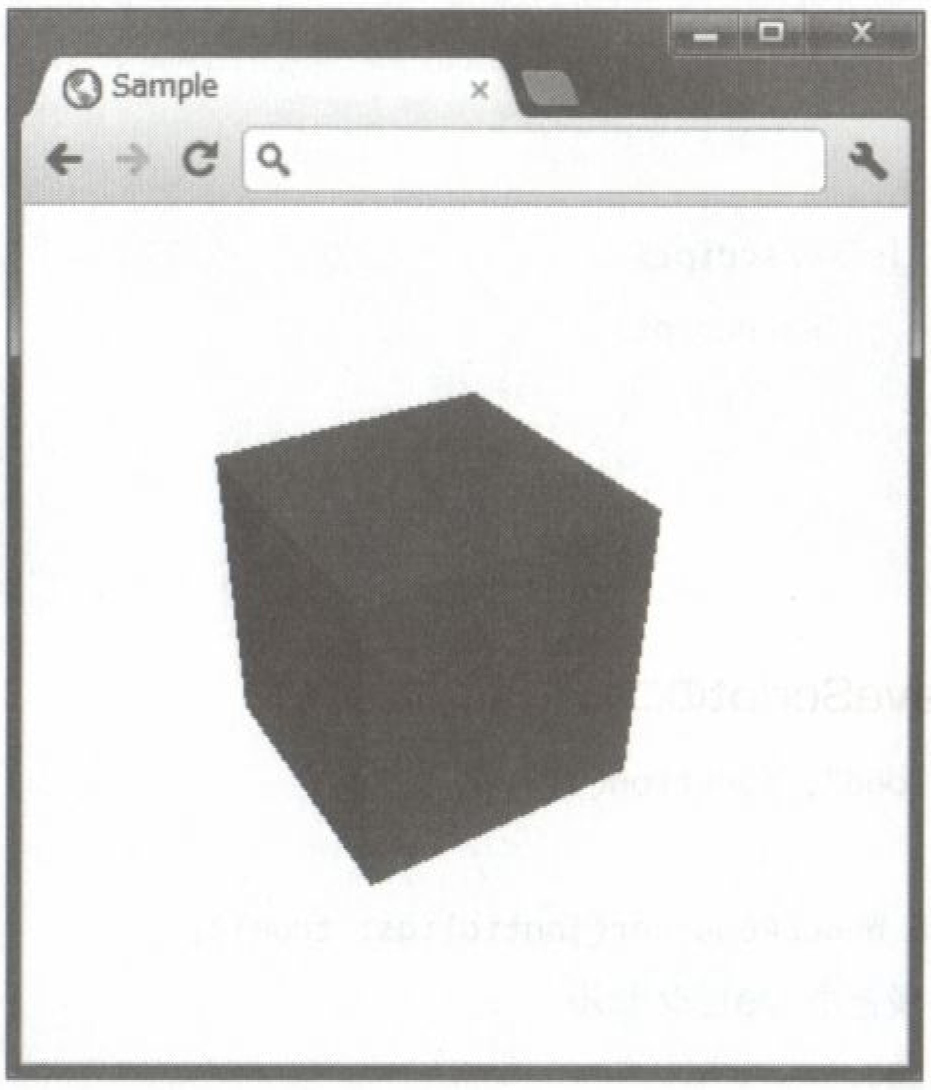
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(20, 20, 20),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x0000ff})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
  // 光源を追加
  var light = new THREE.DirectionalLight(0x0000ff, 1.25);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  // レンダリング
  renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、平行光源が追加され、青色の立方体が描画されます。



ONEPOINT 平行光源を追加するには「new THREE.DirectionalLight()」とする

Three.jsで平行光源を追加するには、「new THREE.DirectionalLight()」とします。最初のパラメータに光源の色を16進数で指定します。2番目のパラメータには、光源の強さを指定します。デフォルトでは、光源の強さは「1.0」になっています。

次に、「position.set()」メソッドを使って、光源の位置を設定します。最後に、「add()」メソッドを使って、描画シーンに光源を追加します。

なお、Three.jsには、環境光・平行光・点光源・スポットライトが用意されています。

コンストラクタ	光源の種類
THREE.AmbientLight()	環境光
THREE.DirectionalLight()	平行光
THREE.PointLight()	点光源
THREE.SpotLight()	スポットライト

関連項目 ▶▶▶

- Three.jsを使って3D処理を行う ……………… p.534
- ワイヤーフレーム表示の幅と不透明度を設定する ……………… p.537
- カメラを設定する ……………… p.539

SECTION-234

立方体を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って立方体を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

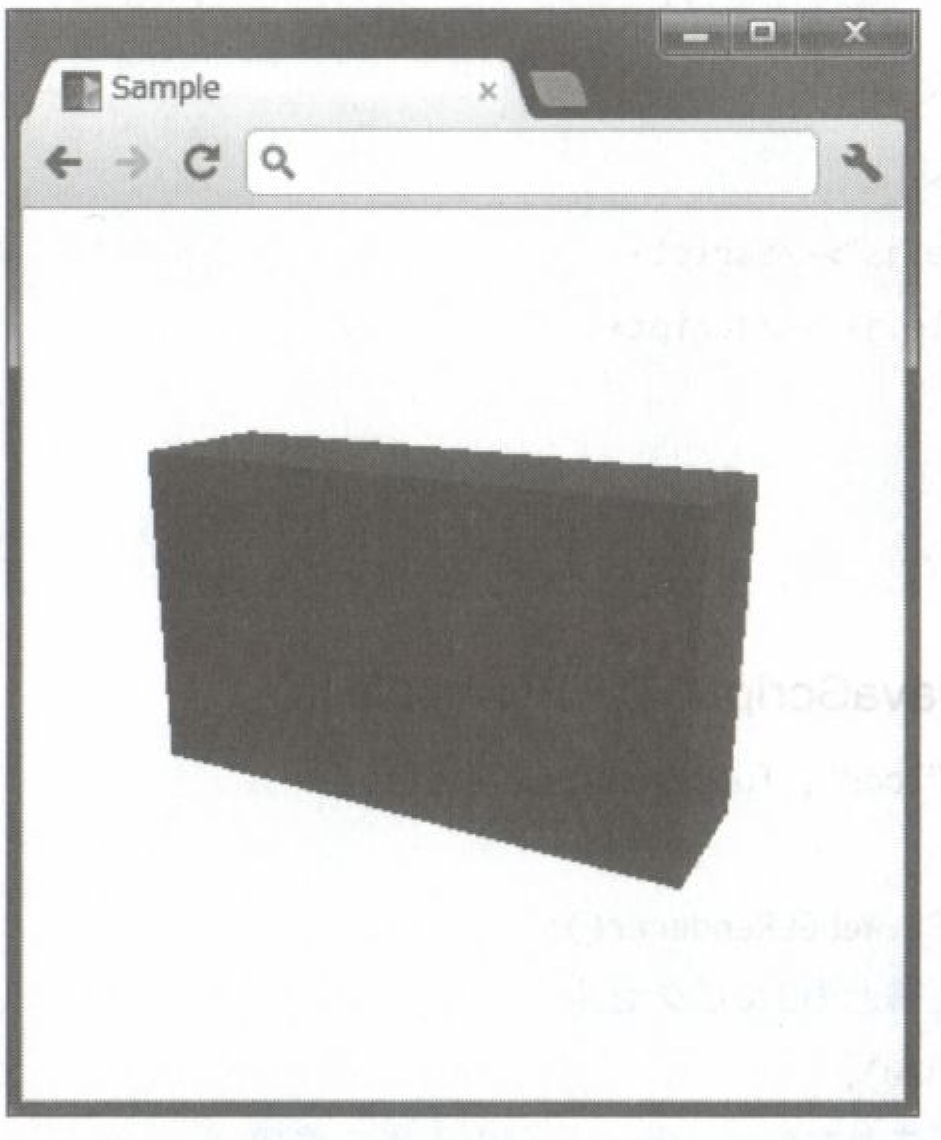
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer({antialias: true});
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(30, 20, 55);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(35, 20, 10),
    // マテリアルをLambertに
    new THREE.MeshPhongMaterial()
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0x0000ff, 1.25);
    // 光源の位置を設定
    light.position.set(200, 260, 100);
    scene.add(light);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、青色の立方体がシーンに追加され、描画されます。



ONEPOINT **立方体を生成するには「new THREE.CubeGeometry()」とする**

Three.jsで立方体を生成するには、「new THREE.CubeGeometry()」とします。パラメータには順番に、横のサイズ、縦のサイズ、高さを指定します。立方体の分割数(セグメント/細かさ)は、4~6番目のパラメータで指定します。これは順番に、横の分割数、縦の分割数、高さの分割数になります。

関連項目 ▶▶▶

- 球体を生成する ……………… p.546
- トーラスを生成する ……………… p.549
- トーラス結び目を生成する ……………… p.551
- 平面を生成する ……………… p.553
- 円筒を生成する ……………… p.555
- 正二十面体を生成する ……………… p.557
- 回転体を生成する ……………… p.560
- 線を生成する ……………… p.563

SECTION-235

球体を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って球体を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

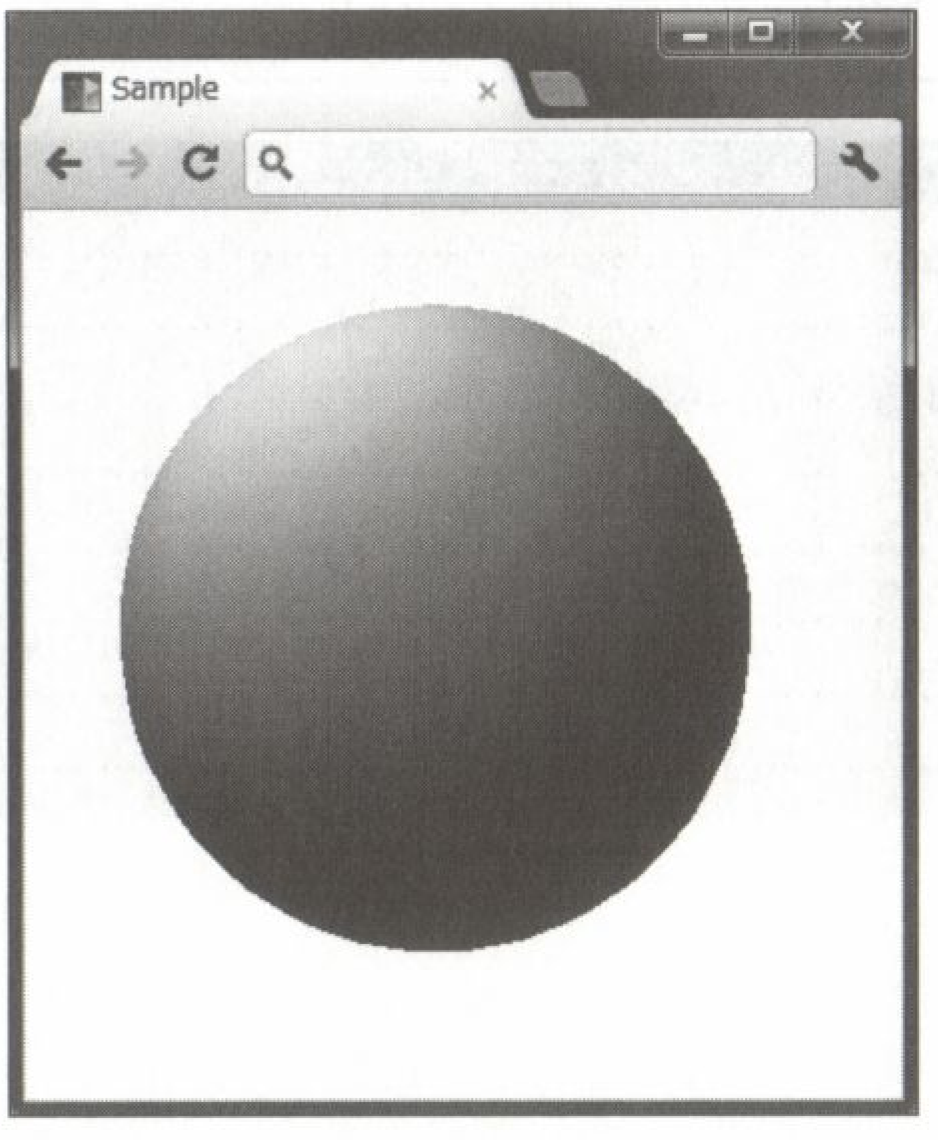
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

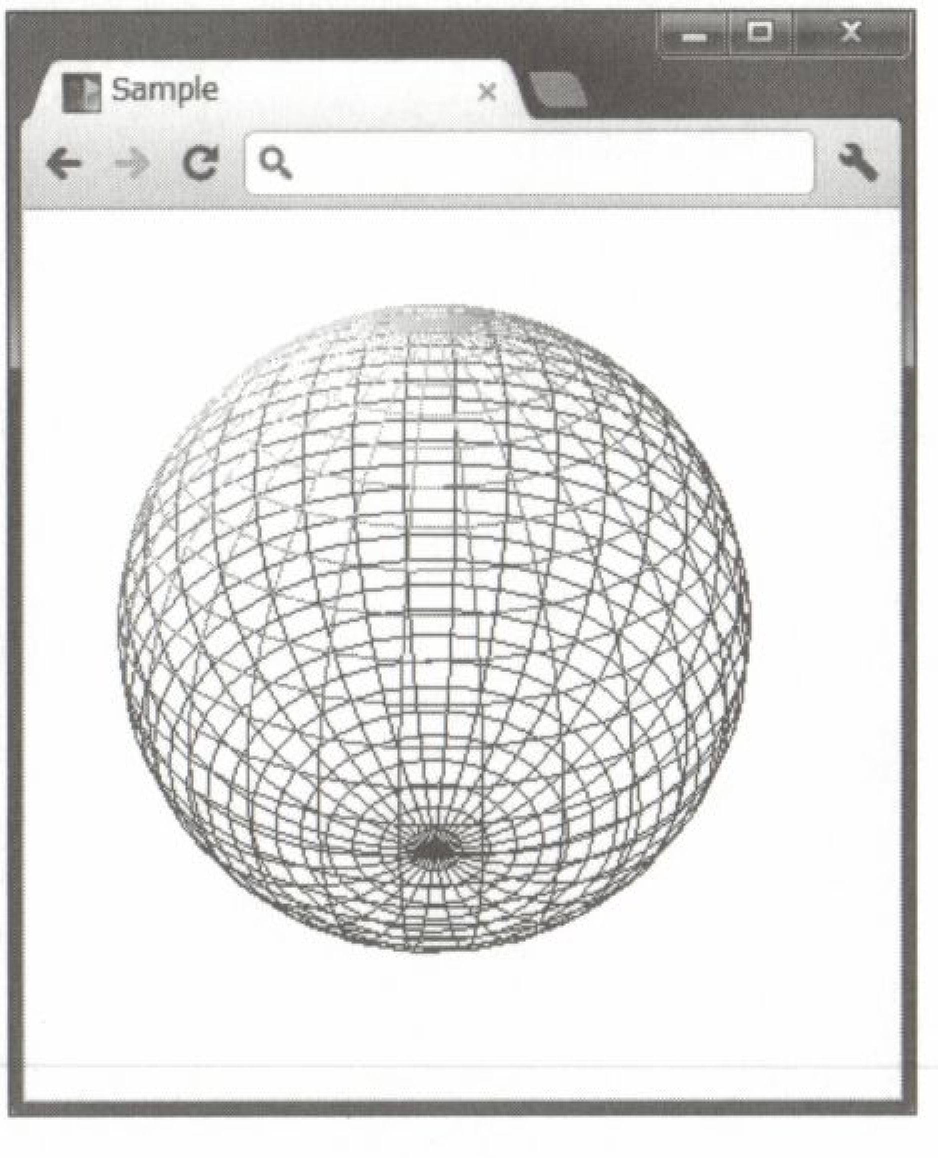
```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var sphere = new THREE.Mesh(
    // 球体のXYZサイズを設定
    new THREE.SphereGeometry(20, 30, 30),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0xffff00})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 球体を3Dシーンに追加
  scene.add(sphere);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
  // 光源を追加
  var light = new THREE.DirectionalLight(0xffffff, 1.0);
  // 光源の位置を設定
  light.position.set(-50, 260, 100);
  scene.add(light);
  // レンダリング
  renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、球体がシーンに追加されて表示されます。



球体の分割数(セグメント)を指定することもできます。ワイヤーフレームで表示すると、どのくらい細かく分割されているか確認できます。



ONEPOINT　球体を生成するには「new THREE.SphereGeometry()」とする

Three.jsで球体を生成するには、「new THREE.SphereGeometry()」とします。最初のパラメータが球体の半径、2番目が横の分割数(セグメント/細かさ)、3番目が縦の分割数になります。いずれのパラメータも省略可能です。

また、完全な球体ではなく、半球などの形状したい場合には、4番目のパラメータにX方向の開始角度、5番目のパラメータにX方向の終了角度を指定します。6番目のパラメータにZ方向の開始角度、7番目のパラメータにZ方向の終了角度を指定します(角度はラジアン単位で指定する)。

関連項目 ▶▶▶

- 立方体を生成する ……………………………… p.544
- トーラスを生成する ……………………………… p.549
- トーラス結び目を生成する ……………………… p.551
- 平面を生成する ………………………………… p.553
- 円筒を生成する ………………………………… p.555
- 正二十面体を生成する ………………………… p.557
- 回転体を生成する ……………………………… p.560
- 線を生成する …………………………………… p.563

SECTION-236

トーラスを生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使ってトーラス(ドーナツ型)を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

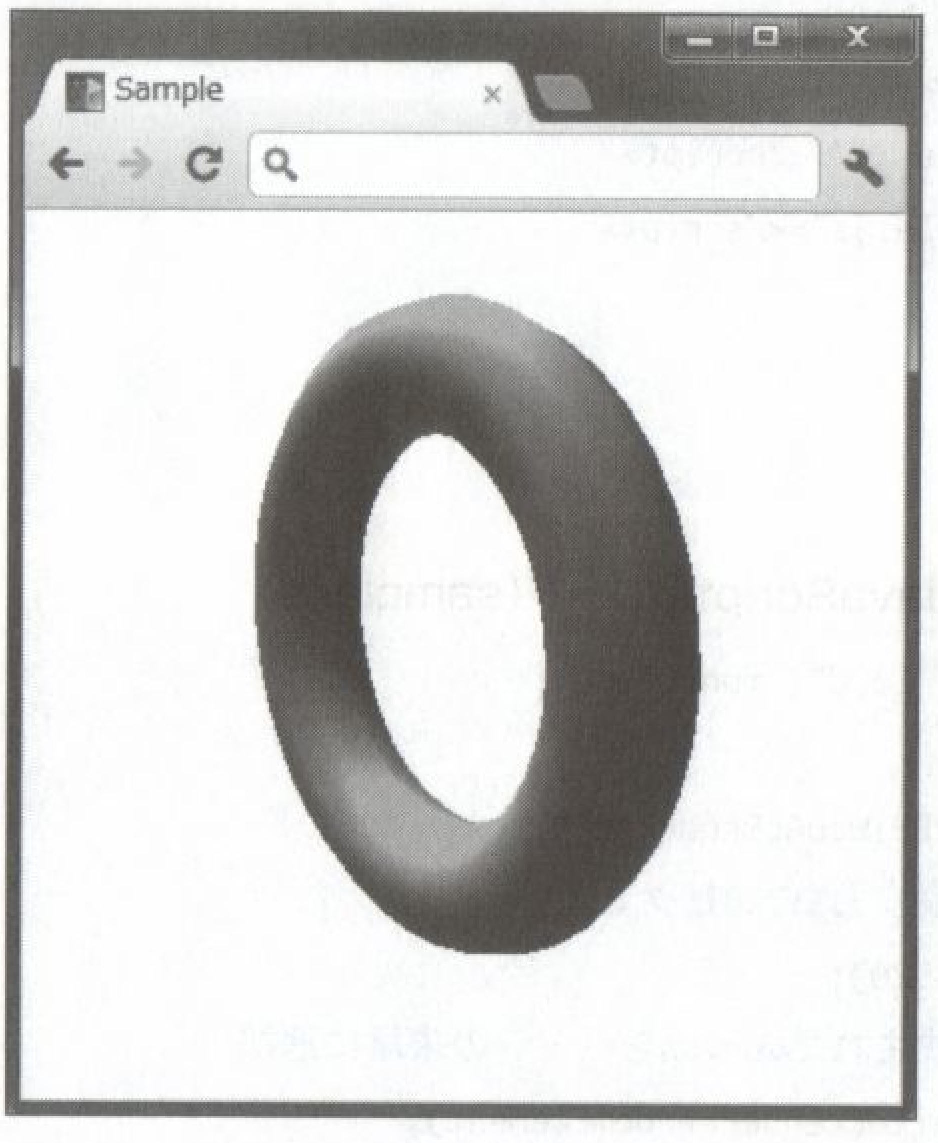
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(40, 10, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var torus = new THREE.Mesh(
    //トーラスのサイズを設定
    new THREE.TorusGeometry(12, 3, 10, 40),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x00ff00})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // トーラスを3Dシーンに追加
  scene.add(torus);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0xffffff, 1.3);
    // 光源の位置を設定
    light.position.set(-50, 260, 100);
    scene.add(light);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、トーラス(ドーナツ型)がシーンに追加されて表示されます。



ONEPOINT トーラスを生成するには「new THREE.TorusGeometry()」とする

Three.jsでトーラス(ドーナツ型)を生成するには、「new THREE.TorusGeometry()」とします。最初のパラメータがトーラスの半径、2番目が管の太さになります。3番目と4番目が分割数(セグメント/細かさ)になります。いずれのパラメータも省略可能です。

また、完全なトーラスではなく、半円などの形状したい場合には、5番目のパラメータに開始角度、6番目のパラメータに終了角度を指定します(角度はラジアン単位で指定する)。

関連項目 ▶▶▶

● 立方体を生成する ………… p.544
● 球体を生成する ………… p.546
● トーラス結び目を生成する ………… p.551
● 平面を生成する ………… p.553
● 円筒を生成する ………… p.555
● 正二十面体を生成する ………… p.557
● 回転体を生成する ………… p.560
● 線を生成する ………… p.563

SECTION-237

トーラス結び目を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使ってトーラス結び目を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

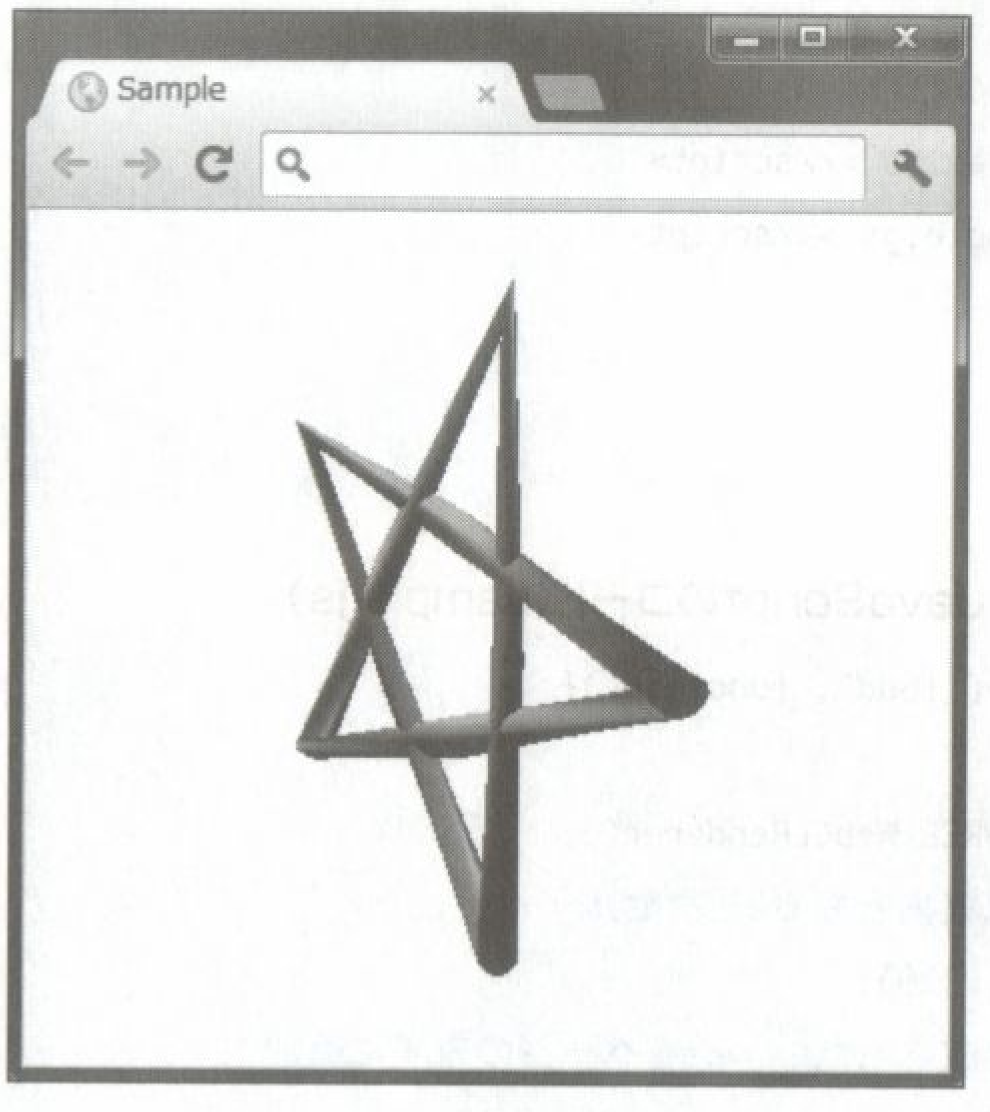
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(40, 10, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var torusknot = new THREE.Mesh(
    // トーラス結び目の形状を設定
    new THREE.TorusKnotGeometry(10, 1, 10, 40, 2, 5),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x00ff00})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // トーラス結び目を3Dシーンに追加
  scene.add(torusknot);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0xffffff, 1.3);
    // 光源の位置を設定
    light.position.set(-50, 260, 100);
    scene.add(light);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、星形のトーラス結び目がシーンに追加されて表示されます。



ONEPOINT トーラス結び目を生成するには「new THREE.TorusKnotGeometry()」とする

Three.jsでトーラス結び目を生成するには、「new THREE.TorusKnotGeometry()」とします。最初のパラメータが半径、2番目が管の太さになります。3番目と4番目が分割数(セグメント/細かさ)になります。いずれのパラメータも省略可能です。

5番目と6番目のパラメータが形状を決めます。Three.jsでは、(p , q)型のトーラス結び目になります。

関連項目 ▶▶▶

- 立方体を生成する ……… p.544
- 球体を生成する ……… p.546
- トーラスを生成する ……… p.549
- 平面を生成する ……… p.553
- 円筒を生成する ……… p.555
- 正二十面体を生成する ……… p.557
- 回転体を生成する ……… p.560
- 線を生成する ……… p.563

SECTION-238

平面を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って平面を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

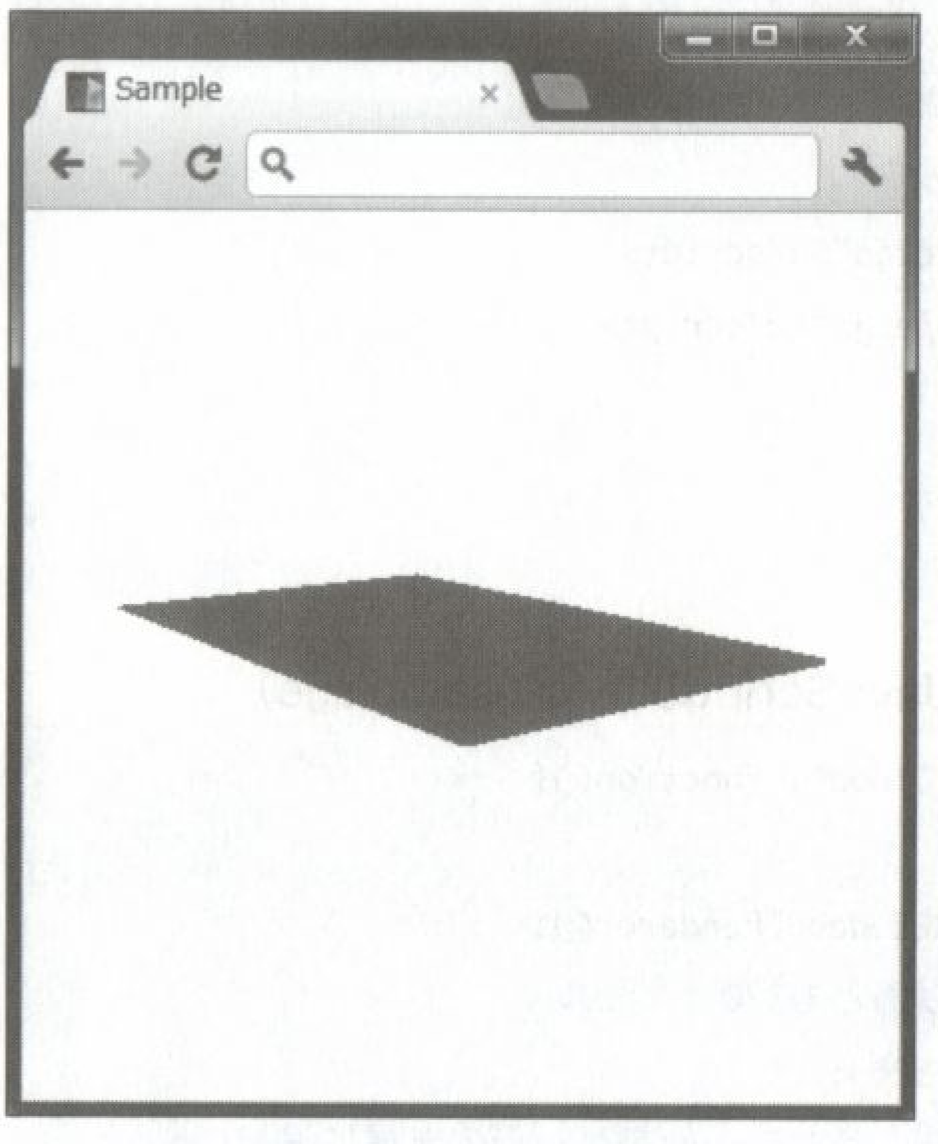
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(40, 10, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var plane = new THREE.Mesh(
    // 平面のサイズを設定
    new THREE.PlaneGeometry(30, 20),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0xff0000})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 平面を3Dシーンに追加
  scene.add(plane);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0xffffff, 0.85);
    // 光源の位置を設定
    light.position.set(-50, 260, 100);
    scene.add(light);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、赤い平面がシーンに追加されて表示されます。



ONEPOINT 平面を生成するには「new THREE.PlaneGeometry()」とする

Three.jsで平面を生成するには、「new THREE.PlaneGeometry()」とします。最初のパラメータが平面の横幅、2番目が平面の縦幅になります。3番目と4番目は平面の分割数(セグメント/細かさ)になります。3番目と4番目のパラメータは省略可能です。

関連項目 ▶▶▶

- 立方体を生成する ………… p.544
- 球体を生成する ………… p.546
- トーラスを生成する ………… p.549
- トーラス結び目を生成する ………… p.551
- 円筒を生成する ………… p.555
- 正二十面体を生成する ………… p.557
- 回転体を生成する ………… p.560
- 線を生成する ………… p.563

SECTION-239

円筒を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って円筒を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

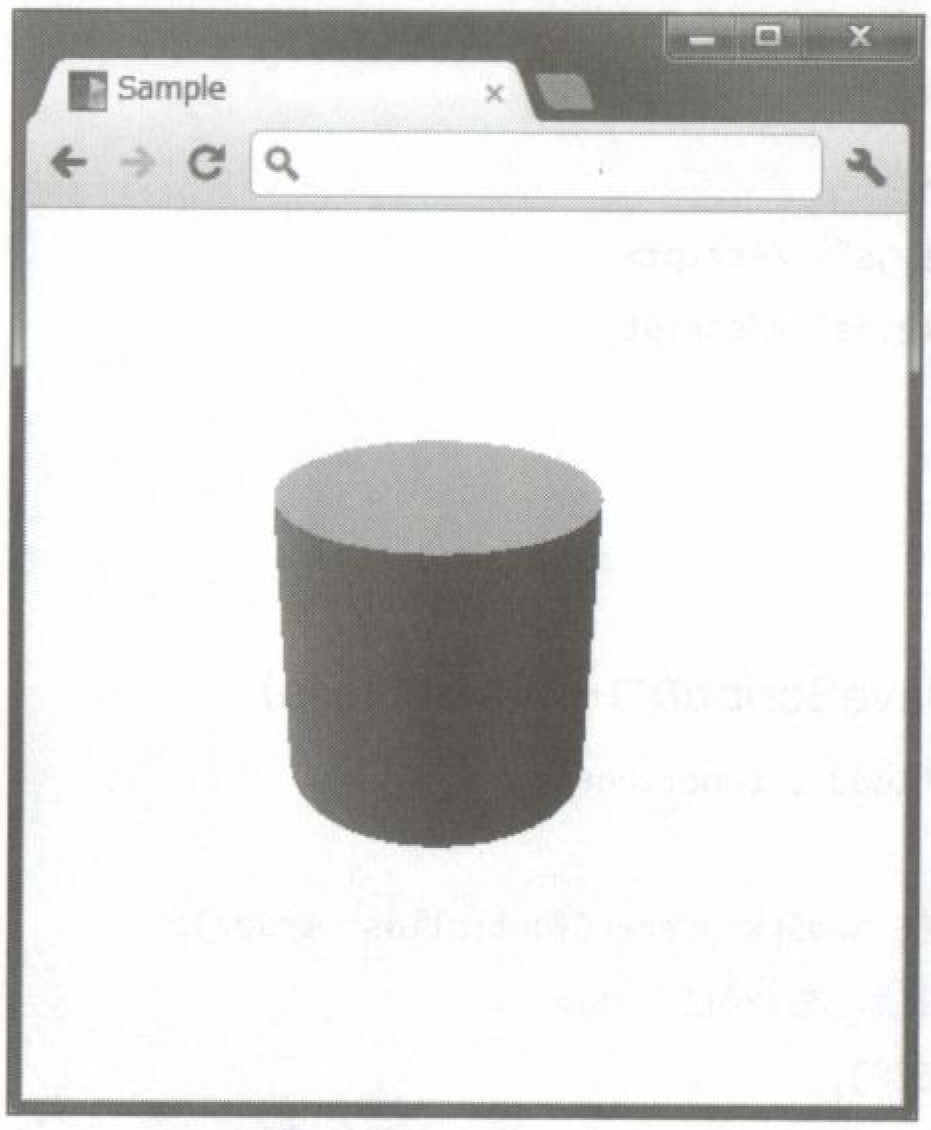
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer({antialias: true});
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(40, 25, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cylinder = new THREE.Mesh(
    // 円筒のサイズを設定
    new THREE.CylinderGeometry(8, 8, 15, 40, 40),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x00ff00})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 円筒を3Dシーンに追加
  scene.add(cylinder);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
        // 光源を追加
        var light = new THREE.DirectionalLight(0xffffff, 1.85);
        // 光源の位置を設定
        light.position.set(-50, 260, 100);
        scene.add(light);
        // レンダリング
        renderer.render(scene, camera);
    }, false);
```

ブラウザでindex.htmlを表示すると、緑色の円筒がシーンに追加されて表示されます。



ONEPOINT 円筒を生成するには「new THREE.CylinderGeometry()」とする

Three.jsで円筒を生成するには、「new THREE.CylinderGeometry()」とします。最初のパラメータが円筒の上の半径、2番目が円筒の下の半径になります。3番目は円筒の高さになります。また、4番目と5番目は平面の分割数(セグメント/細かさ)になります。また、すべてのパラメータは、省略可能です。

関連項目 ▶▶▶

- 立方体を生成する ……… p.544
- 球体を生成する ……… p.546
- トーラスを生成する ……… p.549
- トーラス結び目を生成する ……… p.551
- 平面を生成する ……… p.553
- 正二十面体を生成する ……… p.557
- 回転体を生成する ……… p.560
- 線を生成する ……… p.563

SECTION-240

正二十面体を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って正二十面体を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

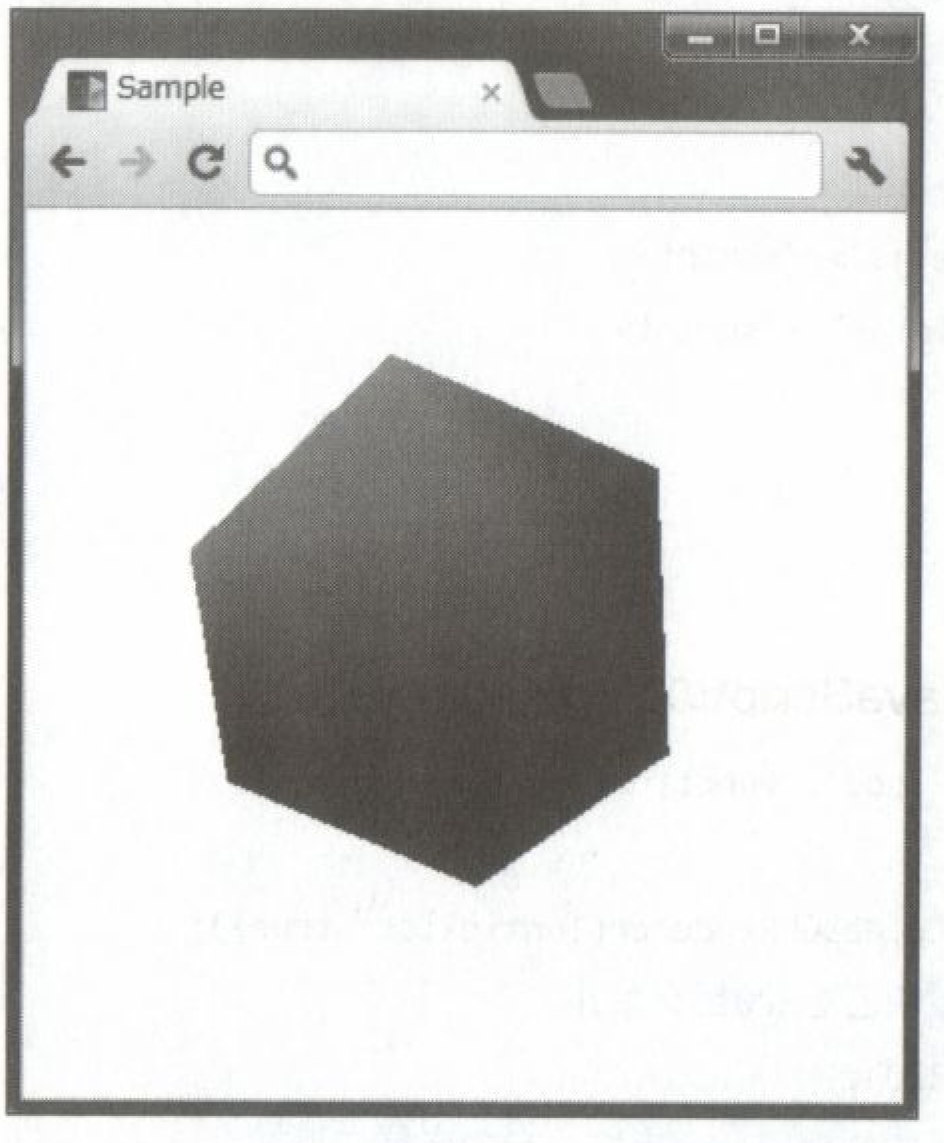
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

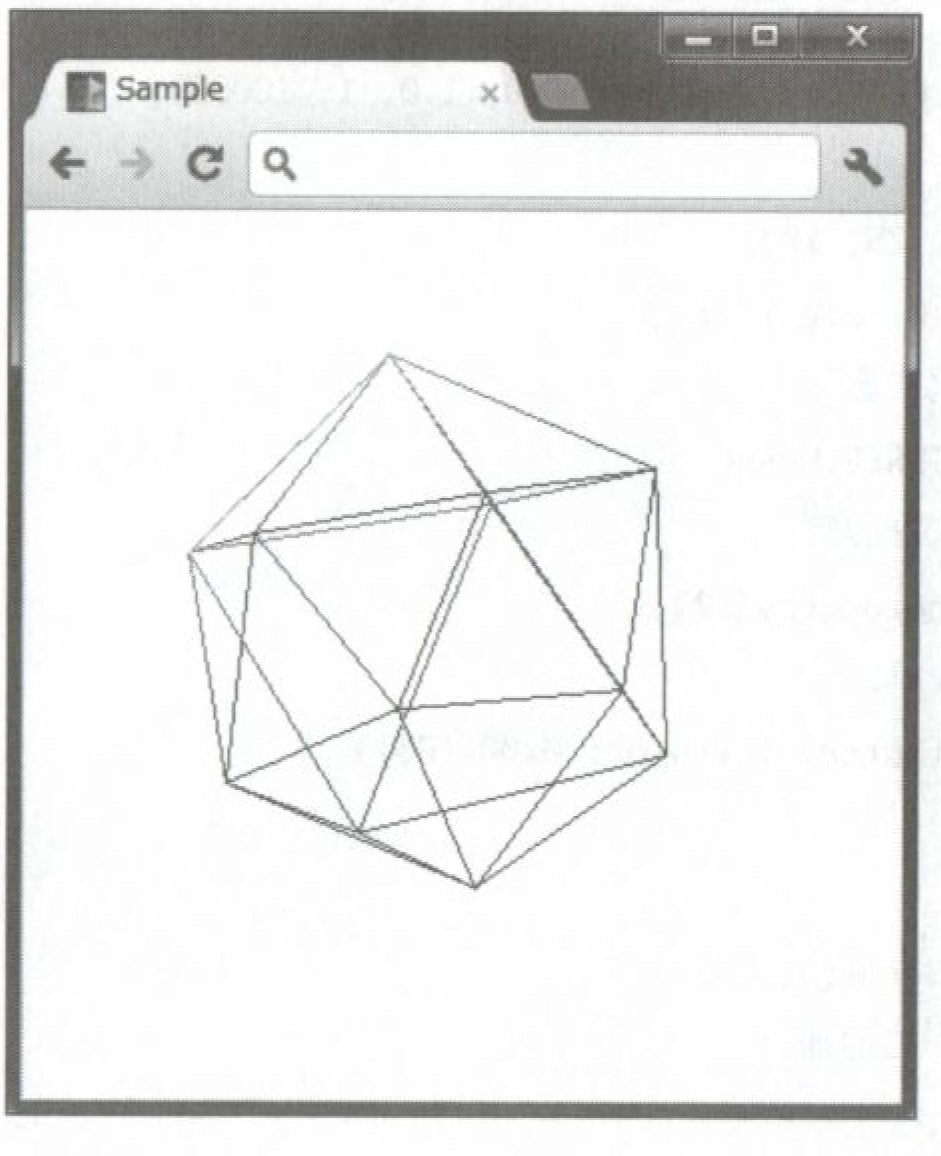
```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer({antialias: true});
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(30, 25, 10);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var icosahedron = new THREE.Mesh(
    // 正二十面体のサイズを設定
    new THREE.IcosahedronGeometry(10),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x00ff00})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 正二十面体を3Dシーンに追加
  scene.add(icosahedron);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0xffffff, 1.0);
    // 光源の位置を設定
    light.position.set(-50, 260, 100);
    scene.add(light);
    // レンダリング
    renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、緑色の正二十面体がシーンに追加されて表示されます。



ワイヤーフレームで表示すると、正二十面体だとわかります。



ONEPOINT **正二十面体を生成するには「new THREE.IcosahedronGeometry()」とする**

Three.jsで正二十面体を生成するには、「new THREE.IcosahedronGeometry()」とします。パラメータには、正二十面体のサイズを指定します。正二十面体の色などは、マテリアルで指定することができます。なお、結果をわかりやすくするため、「wireframe」オプションを指定してワイヤーフレームで表示するサンプルも用意してあります。

関連項目 ▶▶▶

- 立方体を生成する ……………… p.544
- 球体を生成する ……………… p.546
- トーラスを生成する ……………… p.549
- トーラス結び目を生成する ……………… p.551
- 平面を生成する ……………… p.553
- 円筒を生成する ……………… p.555
- 回転体を生成する ……………… p.560
- 線を生成する ……………… p.563

SECTION-241

回転体を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って回転体を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer({antialias: true});
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(30, 35, 70);
  camera.lookAt( {x:10, y:10, z:20 } );
  // ポイントを設定
  var points = [ ];
  points[0] = new THREE.Vector3(0, 0, 0);
  points[1] = new THREE.Vector3(1, 0, 0);
  points[2] = new THREE.Vector3(20, 0, 0);
  points[3] = new THREE.Vector3(2, 0, 10);
  points[4] = new THREE.Vector3(2, 0, 35);
  // オブジェクトを初期化する
  var lathe = new THREE.Mesh(
    // 回転体のサイズを設定
    new THREE.LatheGeometry(points, 6, Math.PI),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x00ff00, wireframe: false})
  );
```

```
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 回転体を3Dシーンに追加
  scene.add(lathe);
  // カメラを3Dシーンに追加
  scene.add(camera);
  // 光源を追加
  var light = new THREE.DirectionalLight(0xffffff, 1.0);
  // 光源の位置を設定
  light.position.set(-50, 260, 100);
  scene.add(light);
  // レンダリング
  renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、緑色の回転体（漏斗を半分に切断した形状）がシーンに追加されて表示されます。



ONEPOINT　回転体を生成するには「new THREE.LatheGeometry()」とする

Three.jsで回転体を生成するには、「new THREE.LatheGeometry()」とします。最初のパラメータには、回転体の元となる座標値を格納した配列を指定します。座標値は、「new THREE.Vector3()」として生成し配列に格納していきます。

回転体を生成する「THREE.LatheGeometry()」の2番目のパラメータには、分割数（セグメント/細かさ）を指定します。3番目のパラメータは、どのくらい回転させるかをラジアン単位で指定します。「Math.PI*2」とすれば360度回転した物体が作成され、「Math.PI」であれば180度だけ回転させた物体が生成されます。「THREE.LatheGeometry()」の2番目以降のパラメータは、省略することができます。

関連項目 ▶▶▶

SECTION-242

線を生成する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って線を生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

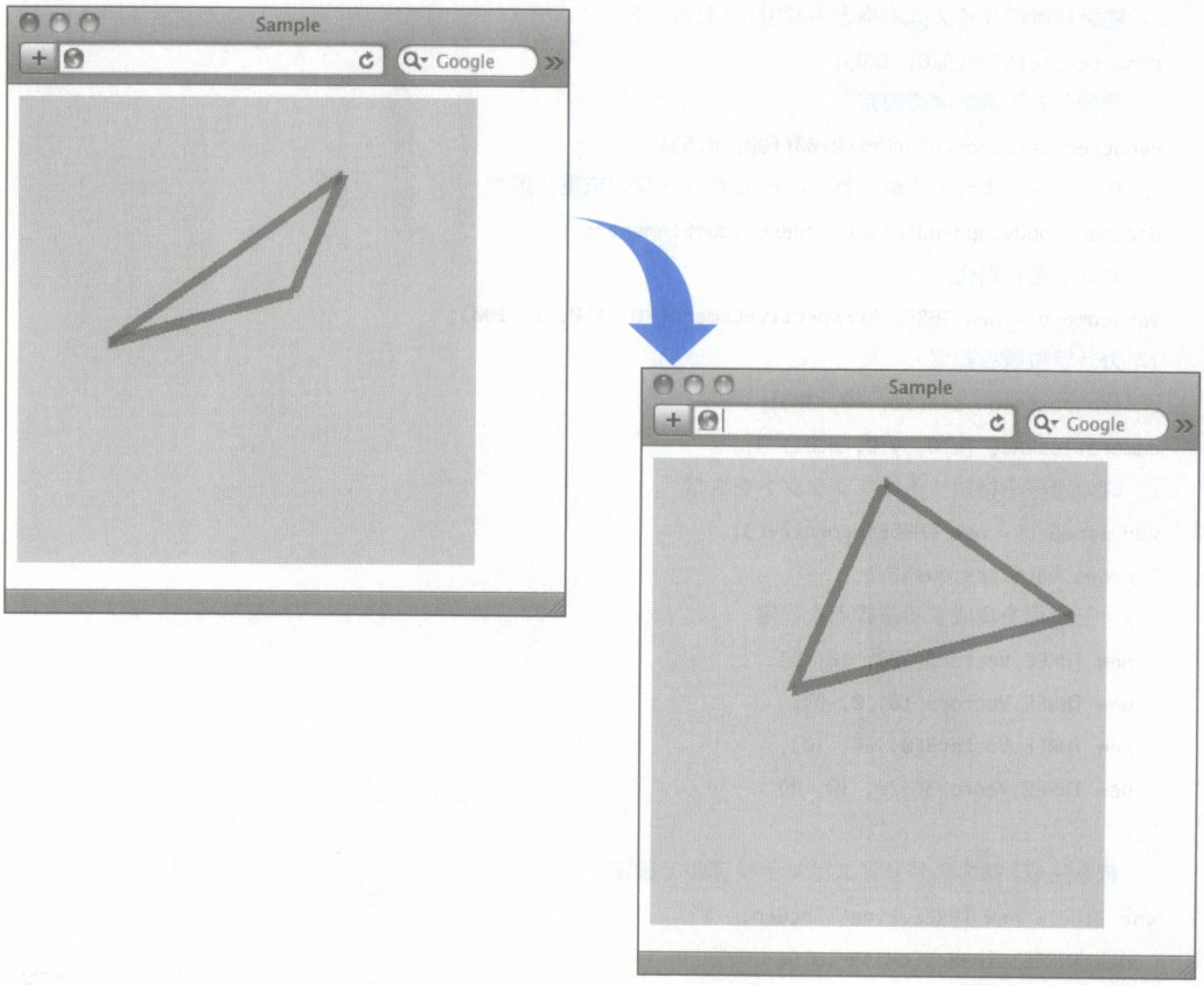
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // 背景色と不透明度を設定
  renderer.setClearColorHex(0x00ff00, 0.5);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 100);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // 線の座標を格納するオブジェクトを生成
  var lineGeo = new THREE.Geometry();
  lineGeo.vertices.push(
    // 三角形を構成する座標点を設定
    new THREE.Vector3(-20, 10, 0),
    new THREE.Vector3(10, 0, 0),
    new THREE.Vector3(0, 20, 10),
    new THREE.Vector3(-20, 10, 0)
  );
  // 線を生成。線の色やサイズはマテリアルで設定
  var line = new THREE.Line(lineGeo,
    new THREE.LineBasicMaterial({
```

▼

```
      color: 0x0000ff,
      linewidth: 8,
      opacity : 0.6
    })
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(line);
  // カメラを追加
  scene.add(camera);
  // レンダリング&アニメーション
  function render(){
    // 線をY軸回転
    line.rotation.y = line.rotation.y + 0.1;
    requestAnimationFrame(render);
    renderer.render(scene, camera);
  }
  render();
}, false);
```

ブラウザでindex.htmlを表示すると、線で構成された三角形がY軸回転するアニメーションが表示されます。



ONEPOINT **線を生成するには「new THREE.Geometry()」とする**

Three.jsで線を生成するには、「new THREE.Geometry()」として線を構成する座標点を用意します。用意した座標点を「THREE.Line()」の最初のパラメータに指定します。2番目のパラメータは、マテリアルの指定になります。線のマテリアルは、「THREE.LineBasicMaterial()」を使って指定します。線の色は「color」プロパティ、線の太さは「linewidth」プロパティで指定します。

なお、Windows 7上では、Google Chrome 21、Firefox 14とも、「linewidth」プロパティの値は反映されず線幅は「1」になります。

関連項目 ▶▶▶

- 立方体を生成する …… p.544
- 球体を生成する …… p.546
- トーラスを生成する …… p.549
- トーラス結び目を生成する …… p.551
- 平面を生成する …… p.553
- 円筒を生成する …… p.555
- 正二十面体を生成する …… p.557
- 回転体を生成する …… p.560

SECTION-243

テクスチャマッピングを行う

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使ってテクスチャマッピングを行う（画像を貼り付ける）方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

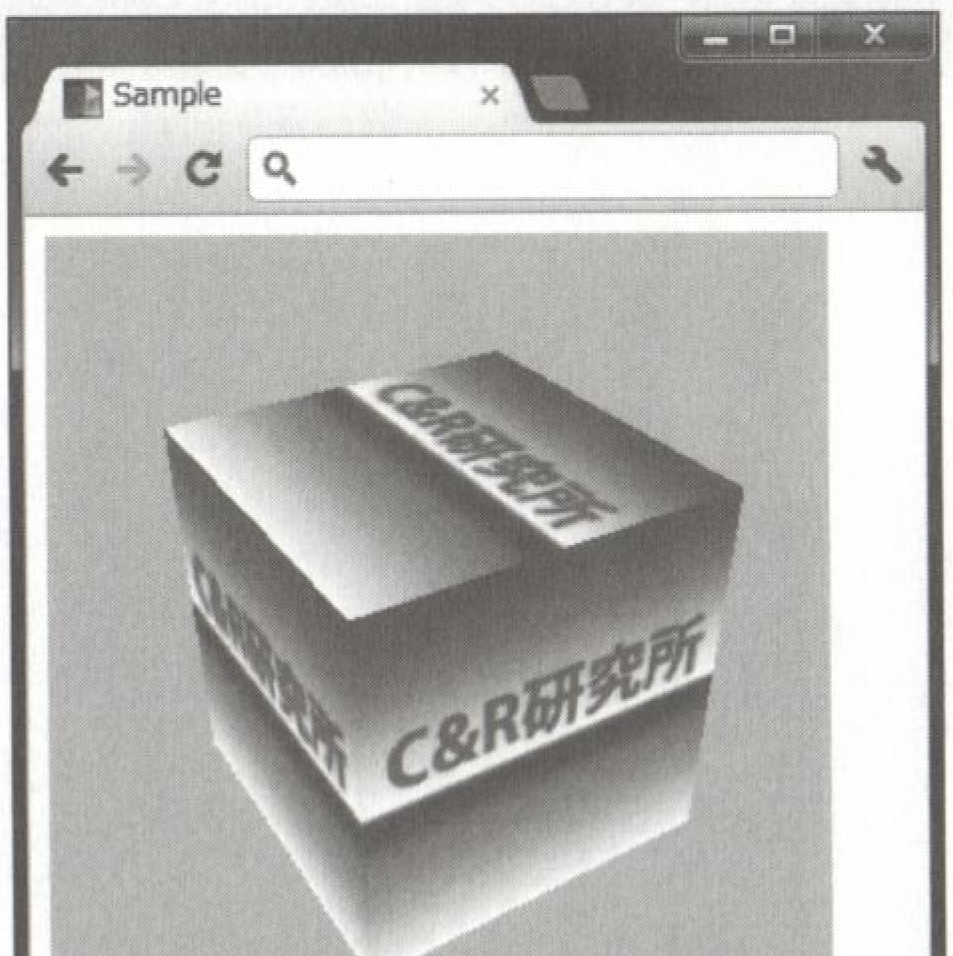
```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // var renderer = new THREE.CanvasRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // 背景色と不透明度を設定
  renderer.setClearColorHex(0x00ff00, 0.5);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 100);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // テクスチャマッピングの画像を読み込む
  var texture  = new THREE.ImageUtils.loadTexture("images/cr.jpg");
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(25, 25, 25),
    // マテリアルをテクスチャに
    new THREE.MeshBasicMaterial({
      map: texture,
      overdraw: true  // THREE.CanvasRendererの場合に必要
    })
```

```
);
// シーンを初期化する
var scene = new THREE.Scene();
// 立方体を3Dシーンに追加
scene.add(cube);
scene.add(camera);
// レンダリング。時間差で2回呼び出さないとテクスチャは表示されない
function render(){
  requestAnimationFrame(render);
  renderer.render(scene, camera);
}
render();
}, false);
```

HINT

画像がローカルディスク上にあると、画像が読み込まれず、真っ黒になる場合があります。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示すると、立方体に画像が貼り付いた状態で描画されます。



ONEPOINT 画像を貼り付けるには「new THREE.ImageUtils.loadTexture()」を使う

Three.jsで立方体や球体などのオブジェクト(3Dモデル)に画像を貼り付ける(テクスチャマッピングを行う)には、「new THREE.ImageUtils.loadTexture()」を使います。パラメータに、貼り付ける画像のURLを指定します。画像がローカルディスク上にあると、画像が読み込まれず、オブジェクトが真っ黒になる場合があります。

読み込まれた画像を貼り付けるには、「new THREE.Mesh()」の2番目のパラメータに、テクスチャを割り当てたマテリアル(質感)を「new THREE.MeshBasicMaterial()」を使って指定します。「new THREE.MeshBasicMaterial()」のパラメータで指定する「map」オプションに「new THREE.ImageUtils.loadTexture()」によって生成した(読み込んだ画像)オブジェクトを指定します。

レンダラーが「THREE.CanvasRenderer」の場合は、これでテクスチャが表示されます。「THREE.WebGLRenderer()」の場合は、「requestAnimationFrame()」を使って2回以上、描画する必要があります(Three.js r49)。

関連項目 ▶▶▶

● 周りの景色を反射させて表示する …………………………………… p.569

SECTION-244

周りの景色を反射させて表示する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って周りの景色を反射させ表示する（環境マップを設定する）方法について解説します。

SAMPLE CODE HTMLのコード（index.html）

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード（sample.js）

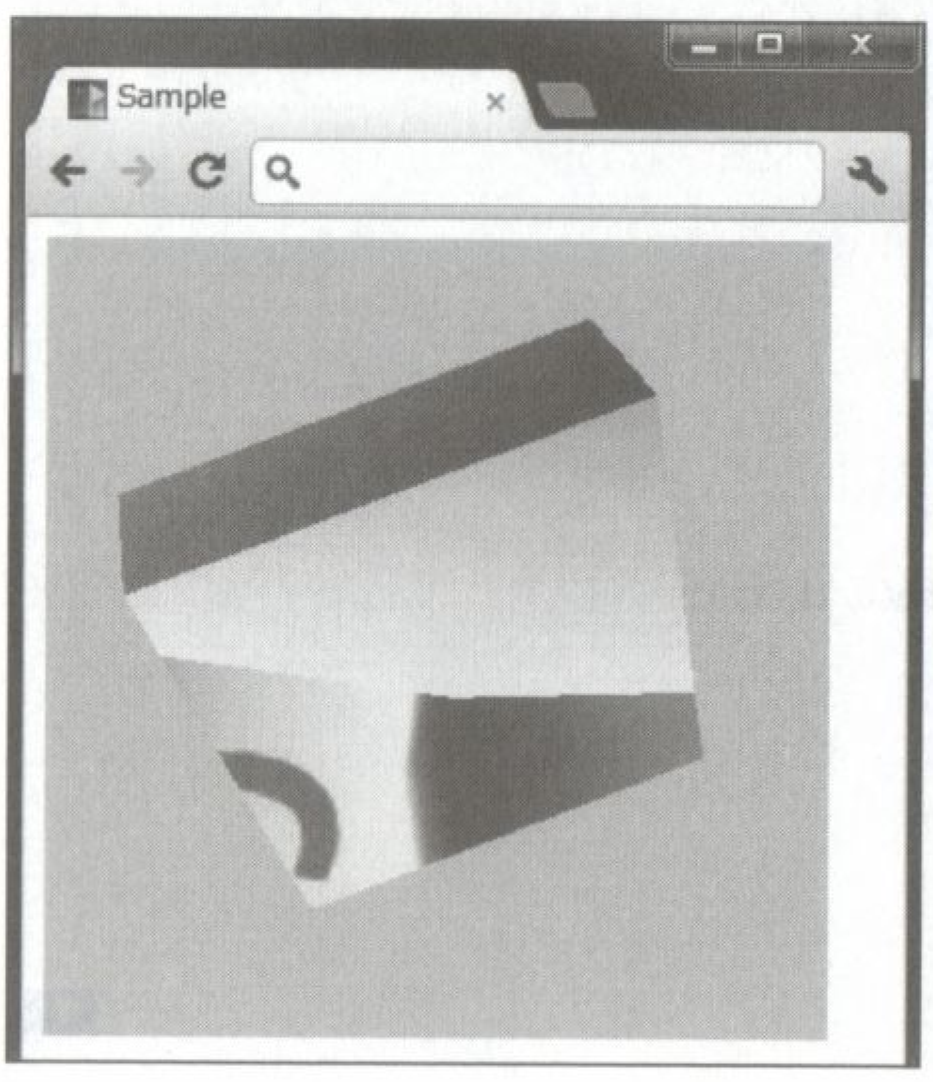
```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // 背景色と不透明度を設定
  renderer.setClearColorHex(0x00ff00, 0.5);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 100);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // テクスチャ(環境)マッピングの画像を読み込む
  var urlList = [
    "images/cr.jpg", "images/cr.jpg",
    "images/cr.jpg", "images/cr.jpg",
    "images/cr.jpg", "images/cr.jpg"
  ];
  var texture  = new THREE.ImageUtils.loadTextureCube(urlList);
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(10, 30, 30),
```

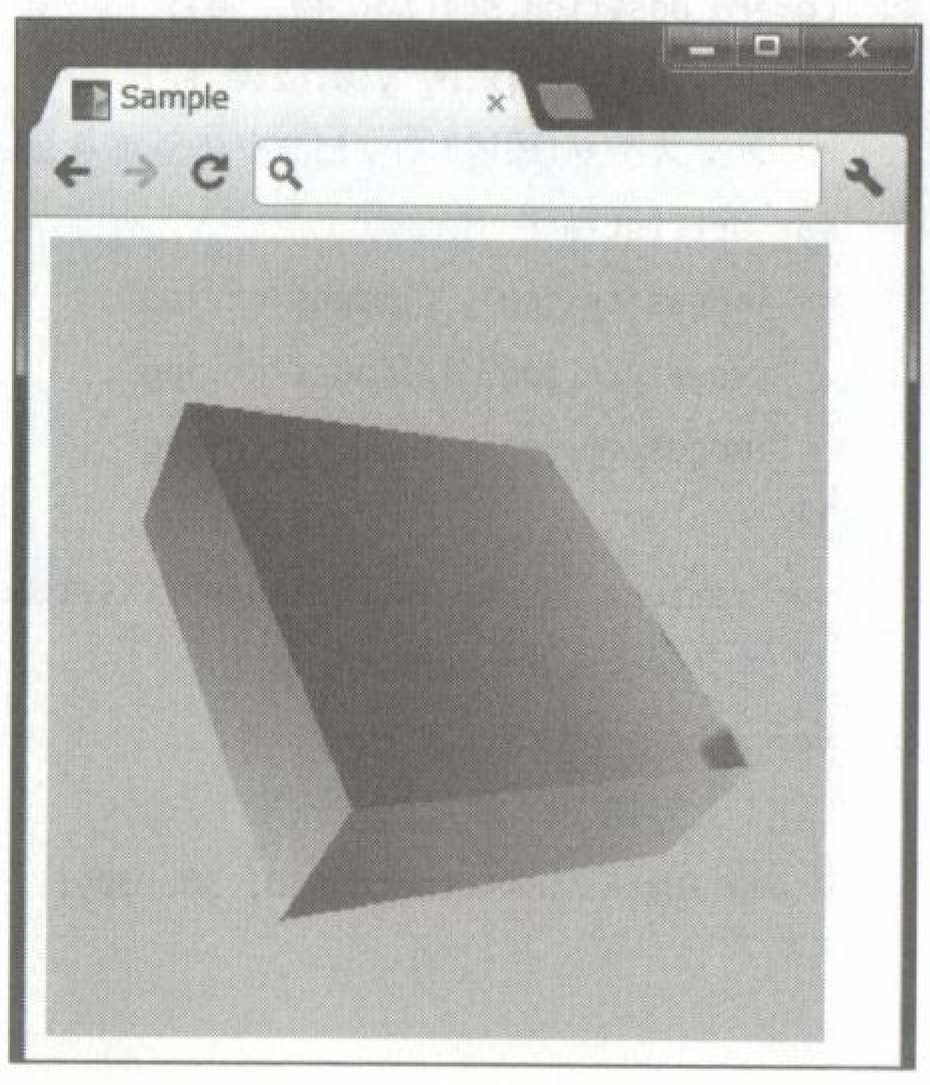
```
    // 環境マップをマテリアルに設定
    new THREE.MeshBasicMaterial({
      color: 0xffff00,
      envMap: texture
    })
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラをシーンに追加
  scene.add(camera);
  // 揺動させるための角度を入れる変数
  var degree = 0;
  // レンダリング
  function render(){
    // 立方体を回転
    degree = degree + 1;
    cube.rotation.x = degree * Math.PI / 180;
    cube.rotation.y = degree * Math.PI / 180;
    requestAnimationFrame(render);
    renderer.render(scene, camera);
  }
  render();
}, false);
```

HINT

画像がローカルディスク上にあると、画像が読み込まれず、真っ黒になる場合があります。その場合は、ネットワーク環境（Webサーバー上）で確認してください。

ブラウザでindex.htmlを表示し、立方体が回転すると、設定された環境マップが適用され反射したように表示されます。





ONEPOINT 環境マップを設定するには「THREE.ImageUtils.loadTextureCube()」を使う

Three.jsで物体に周りの景色を反射させるには、マテリアルの設定で「envMap」プロパティに反射表示させるテクスチャを設定します。このテクスチャは、「THREE.ImageUtils.loadTextureCube()」を使って指定されたものになります。「THREE.ImageUtils.loadTextureCube()」で設定する反射マップ画像は、立方体の6面に対応したものを指定します。

関連項目 ▶▶▶

● テクスチャマッピングを行う……p.566

SECTION-245

立方体などの物体を回転させる

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って立方体などの物体を回転させる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

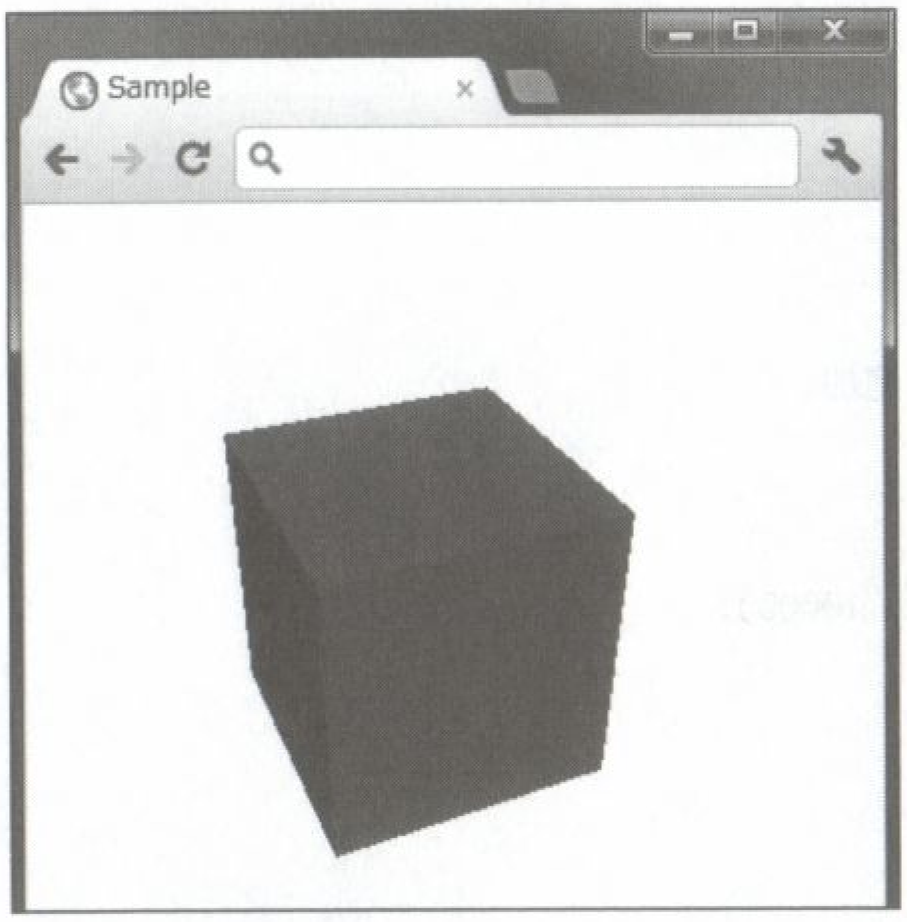
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

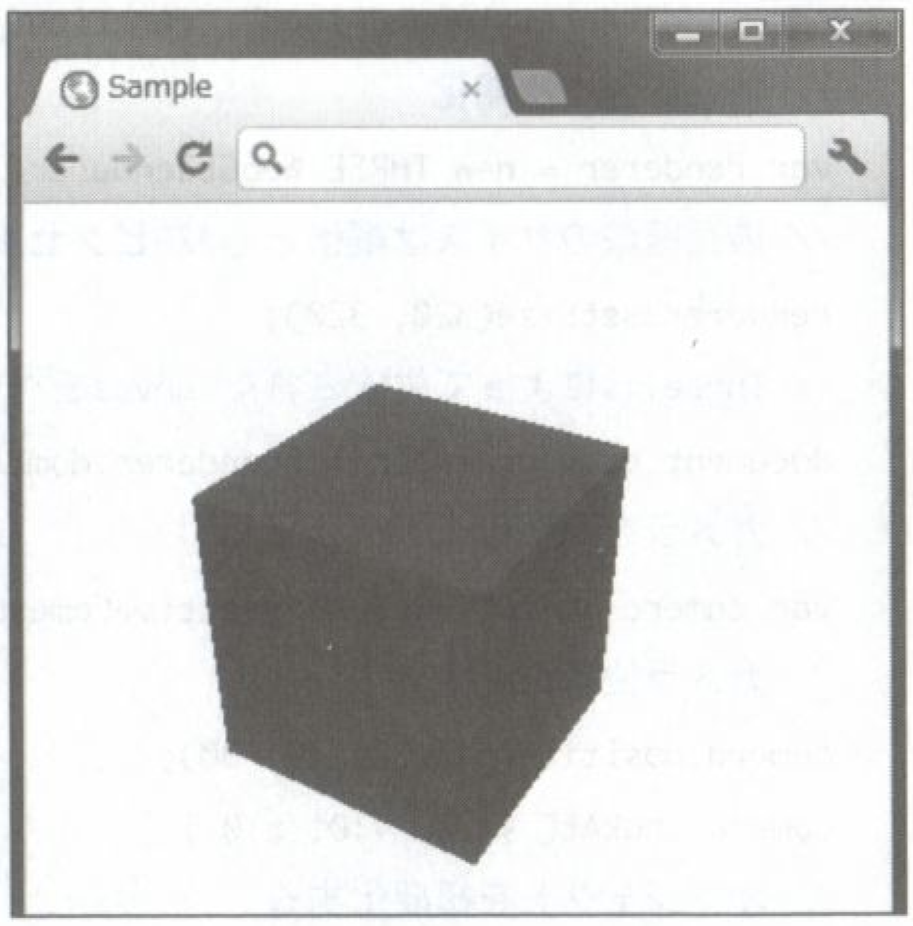
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(20, 20, 20),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x0000ff})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
  // 光源を追加
  var light = new THREE.DirectionalLight(0x0000ff, 1.25);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  var rotY = 0;
  // レンダリング。アニメーションごとに呼び出す
  function render(){
    // 立方体のY軸を回転
    rotY = rotY + 0.01;
    cube.rotation.y = rotY;
    // 定期的に描画するようにする
    requestAnimationFrame(render);
    renderer.render(scene, camera);
  }
  render();
}, false);
```

ブラウザでindex.htmlを表示すると、青色の立方体がY軸回転します。回転角度を指定すれば、他の軸も同時に回転させることもできます。





ONEPOINT **回転させるには「rotation.x」「rotation.y」「rotation.z」プロパティを使う**

Three.jsで作成した物体を回転させるには、物体の情報を持つオブジェクトの「rotation.x」プロパティ、「rotation.y」プロパティ、「rotation.z」プロパティに角度(ラジアン)を設定します。角度を設定しただけではアニメーションしないため、角度を少しずつ変更します。角度を変更したら、一定時間ごと「render()」メソッドを呼び出します。これで、物体がアニメーションするようになります。

関連項目 ▶▶▶

● 立方体などの物体を移動させる……………………………………p.574
● 立方体などの物体のスケールを変更する…………………………p.577

SECTION-246

立方体などの物体を移動させる

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って立方体などの物体を移動させる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

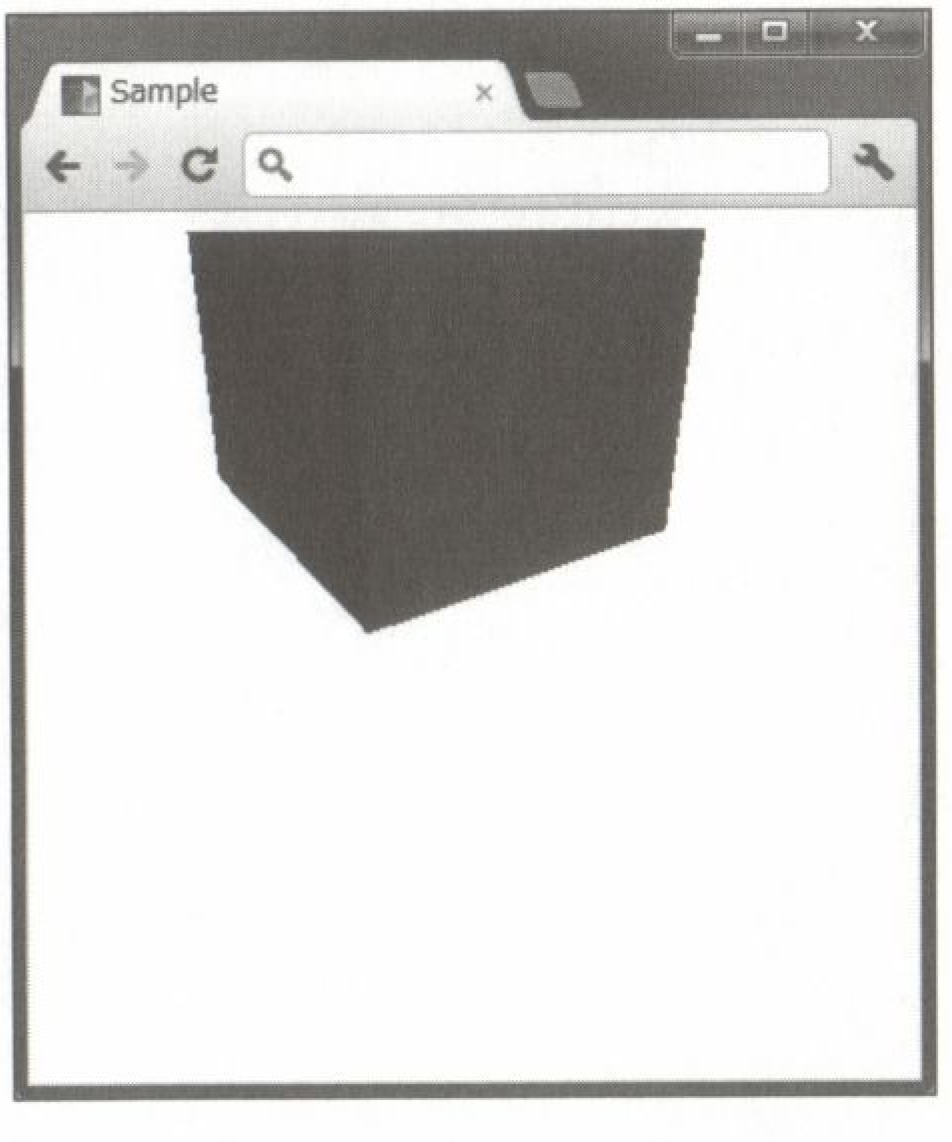
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

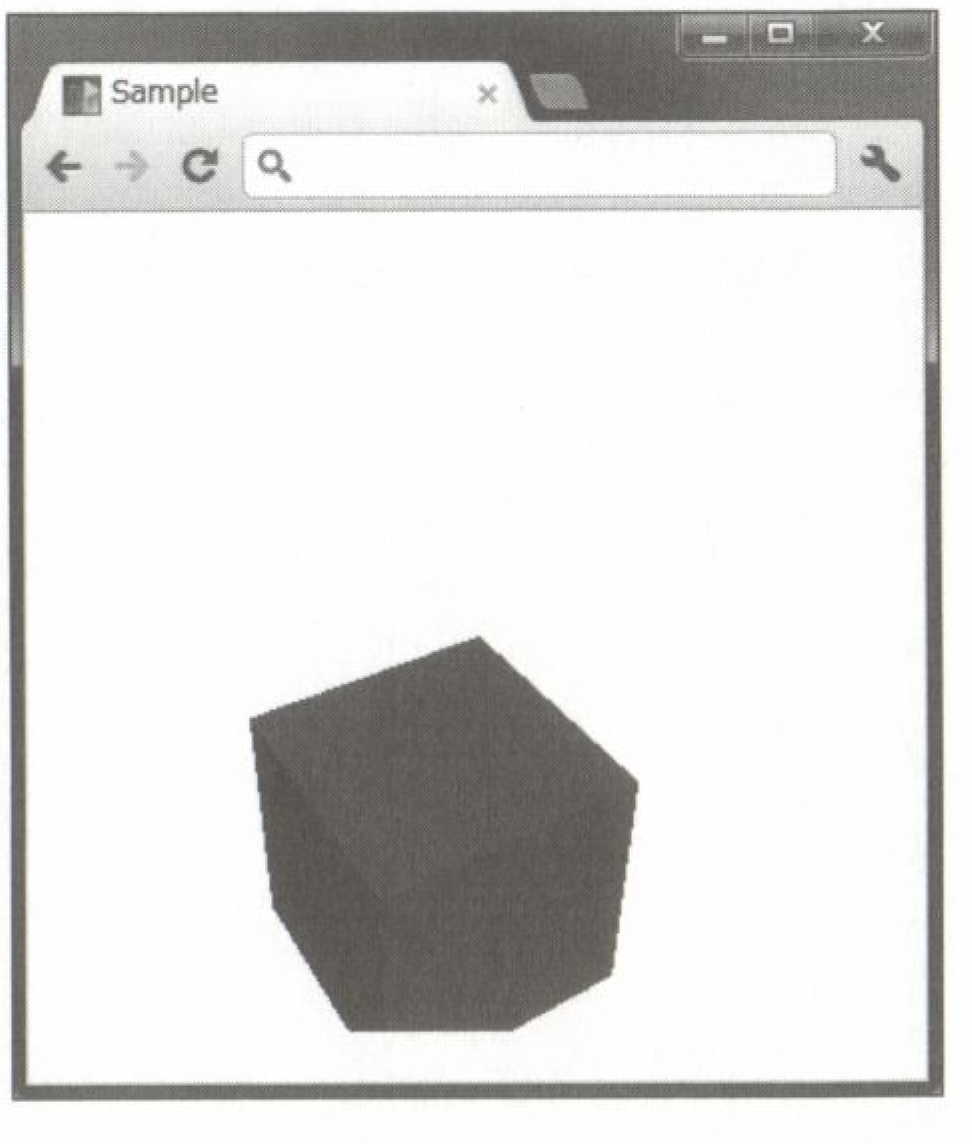
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(20, 20, 20),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x0000ff})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
// 光源を追加
var light = new THREE.DirectionalLight(0x0000ff, 1.25);
// 光源の位置を設定
light.position.set(200, 260, 100);
scene.add(light);
// アニメーションする際に必要な角度を入れておく変数
var degree = 0;
// レンダリング。アニメーションごとに呼び出す
function render(){
  // 立方体のY座標をサインカーブに沿って設定
  degree = degree + 2;
  var rad = degree * Math.PI / 180;
  var posY = Math.sin(rad) * 20;
  cube.position.y = posY;
  // 定期的に描画するようにする
  requestAnimationFrame(render);
  renderer.render(scene, camera);
 }
 render();
}, false);
```

ブラウザでindex.htmlを表示すると、Y方向のみ座標を変更しているため、青色の立方体が上下に動きます。





ONEPOINT 移動させるには「position.x」「position.y」「position.z」プロパティを使う

Three.jsで作成した物体を移動させるには、物体の情報を持つオブジェクトの「position.x」プロパティ、「position.y」プロパティ、「position.z」プロパティに座標値を設定します。座標値を設定しただけではアニメーションしないため、座標値を少しずつ変更します。座標値を変更したら、一定時間ごとに「render()」メソッドを呼び出します。これで、物体がアニメーションして移動するようになります。

関連項目 ▶▶▶

- 立方体などの物体を回転させる……p.572
- 立方体などの物体のスケールを変更する……p.577

SECTION-247

立方体などの物体のスケールを変更する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って立方体などの物体のスケールを変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

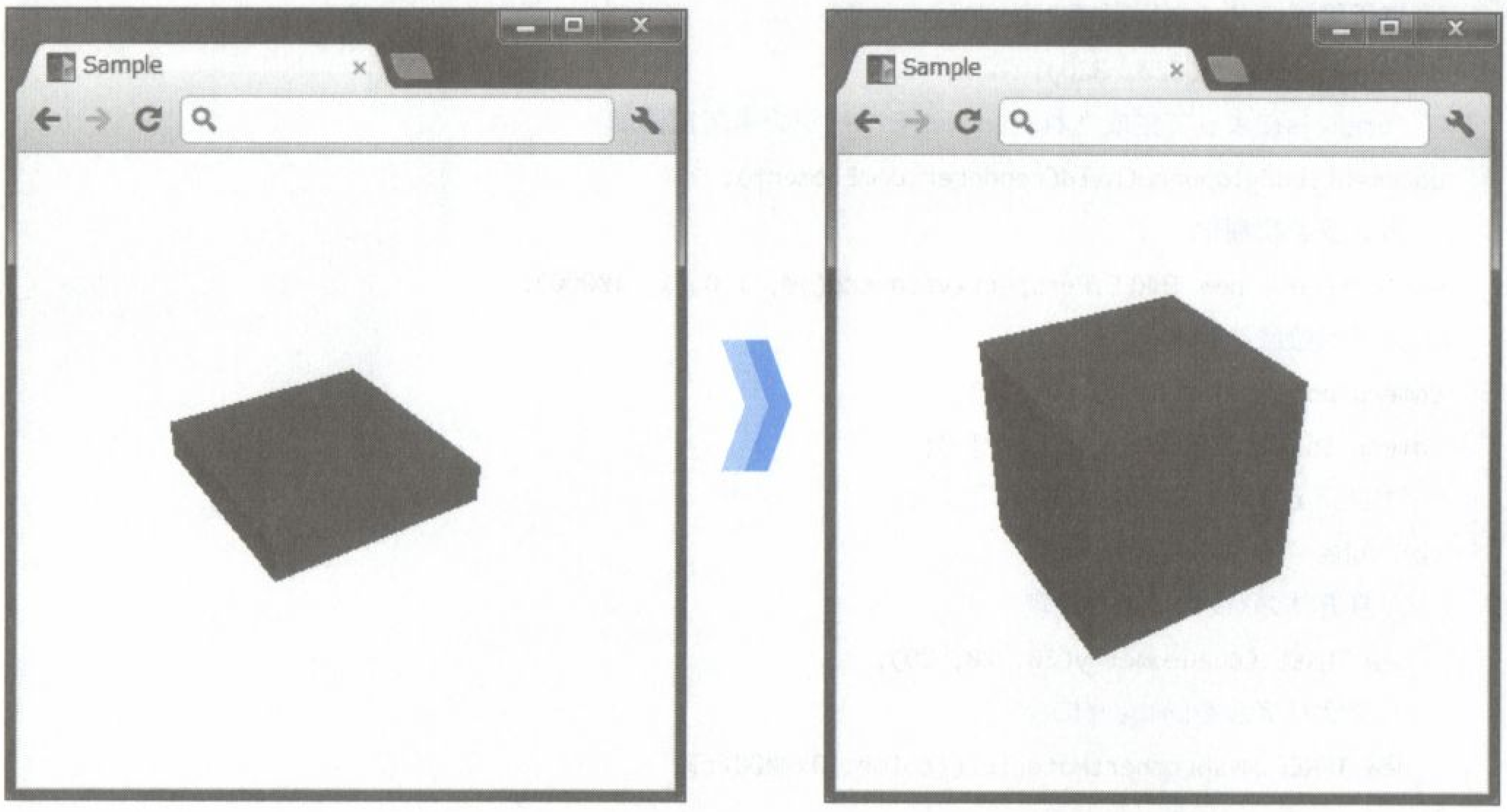
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var cube = new THREE.Mesh(
    // 立方体のXYZサイズを設定
    new THREE.CubeGeometry(20, 20, 20),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial({color: 0x0000ff})
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // 立方体を3Dシーンに追加
  scene.add(cube);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
    // 光源を追加
    var light = new THREE.DirectionalLight(0x0000ff, 1.25);
    // 光源の位置を設定
    light.position.set(200, 260, 100);
    scene.add(light);
    // アニメーションする際に必要な角度を入れておく変数
    var degree = 0;
    // レンダリング。アニメーションごとに呼び出す
    function render(){
      // 立方体の長さをサインカーブに沿って設定
      degree = degree + 2;
      var rad = degree * Math.PI / 180;
      var scaleY = Math.abs(Math.sin(rad));
      cube.scale.y = scaleY;
      // 定期的に描画するようにする
      requestAnimationFrame(render);
      renderer.render(scene, camera);
    }
    render();
  }, false);
```

ブラウザでindex.htmlを表示すると、立方体が縦方向に伸縮するアニメーションが表示されます。



ONEPOINT スケールを変更するには「scale.x」「scale.y」「scale.z」プロパティを使う

Three.jsで作成した物体のスケールを変更するには、物体の情報を持つオブジェクトの「scale.x」プロパティ、「scale.y」プロパティ、「scale.z」プロパティにスケール値を設定します。「1.0」が100%サイズになります。「2.0」を指定すると、2倍のサイズになります。また、スケール値を設定しただけではアニメーションしないため、値を少しずつ変更します。値を変更したら、一定時間ごとに「render()」メソッドを呼び出します。これで、物体の大きさが変化するアニメーションが表示されます。

関連項目 ▶▶▶

- 立方体などの物体を回転させる……p.572
- 立方体などの物体を移動させる……p.574

文字を3Dで描画する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って文字を3Dで描画する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

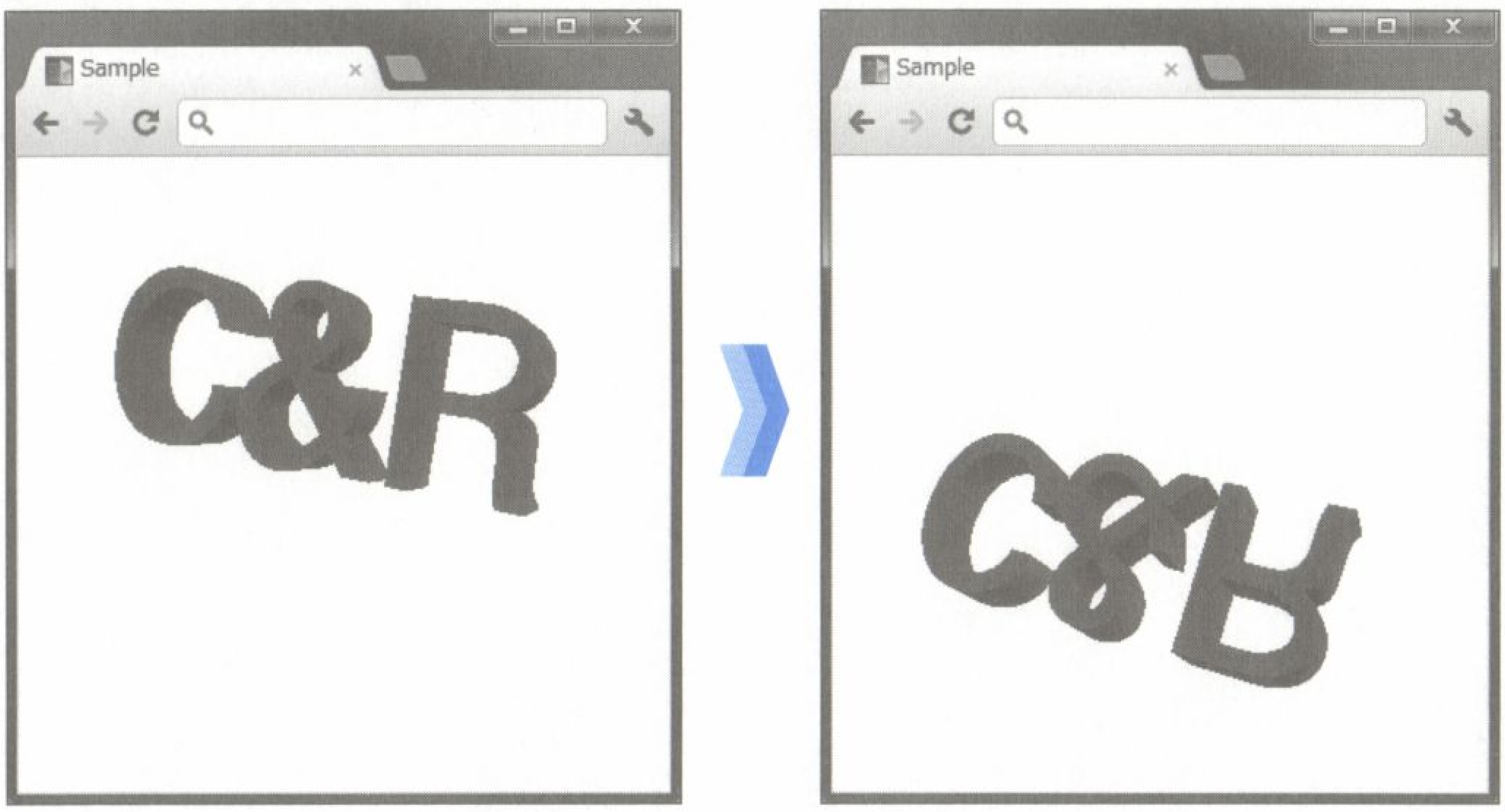
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/helvetiker_regular.typeface.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 30, 60);
  camera.lookAt( {x:20, y:0, z:0 } );
  // テキストを初期化する
  var text3D = new THREE.Mesh(
    // 文字データの設定
    new THREE.TextGeometry(
      // 描画する文字を設定
      "C&R", {
      // サイズと高さを描画
      size: 15,
      height: 6
    }),
    // マテリアルの設定(スムースシェーディングを指定)
    new THREE.MeshPhongMaterial({ color: 0xffffff, shading: THREE.SmoothShading })
  );
```

```
  // シーンを初期化する
  var scene = new THREE.Scene();
  // テキストを3Dシーンに追加
  scene.add(text3D);
  // カメラを3Dシーンに追加
  scene.add(camera);
  // 光源を追加
  var light = new THREE.DirectionalLight(0x0000ff, 2);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  // アニメーションする際に必要な角度を入れておく変数
  var rotX = 0;
  // レンダリング。アニメーションごとに呼び出す
  function render(){
    // 文字のY軸を回転
    rotX = rotX + 0.02;
    text3D.rotation.x = rotX;
    // 定期的に描画するようにする
    requestAnimationFrame(render);
    renderer.render(scene, camera);
  }
  render();
}, false);
```

ブラウザでindex.htmlを表示すると、C&Rの文字が表示され、X軸に回転します。



ONEPOINT Three.jsで文字を描画するにはtypeface.jsを使う

Three.jsで文字を描画するには、typeface.jsが必要です。Three.jsファイルを読み込ませた後に使用する書体に応じたtypeface.jsライブラリファイルを読み込ませます。

次に「new THREE.Mesh()」を使って描画する文字とマテリアル(質感)を指定します。最初のパラメータには、「new THREE.TextGeometry()」を指定します。パラメータには、文字とサイズや奥行きの幅を指定します。文字のサイズは「size」、奥行きの幅は「height」で指定します。「new THREE.Mesh()」の2番目のパラメータには、マテリアルを指定します。

なお、typeface.jsは次のURLから入手することができます。

- typeface.js

  URL http://typeface.neocracy.org/

COLUMN ベベルを指定するには

文字にベベルを指定する場合は、「THREE.TextGeometry()」に、次のオプションを指定します。

```
bevelEnabled : true,
bevelSize : 0.5,
bevelThickness : 1
```

SECTION-249

霧(フォグ)を設定する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って霧(フォグ)を設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

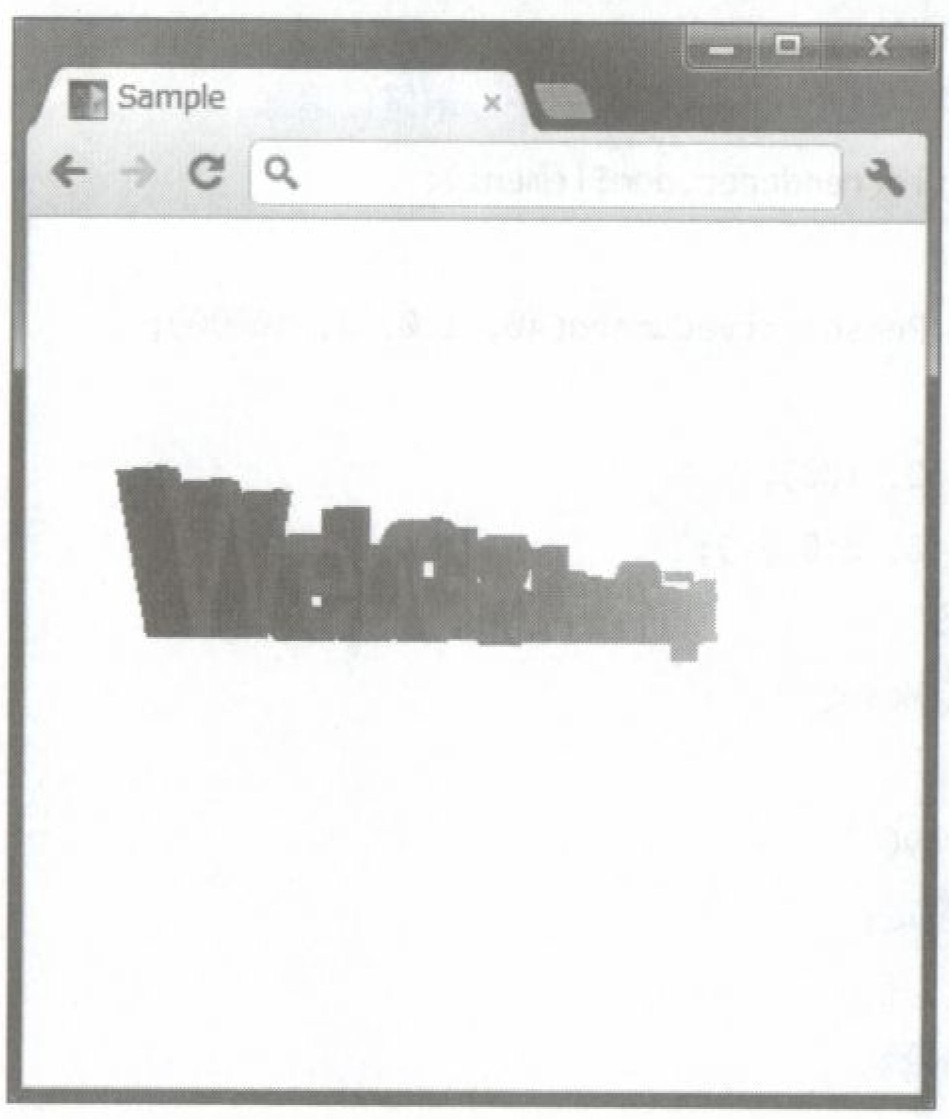
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/helvetiker_regular.typeface.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(0, 0, 100);
  camera.lookAt( {x:0, y:0, z:0 } );
  // テキストを初期化する
  var text3D = new THREE.Mesh(
    // 文字データの設定
    new THREE.TextGeometry(
      // 描画する文字を設定
      "WebGL&JavaScript", {
      // サイズと高さを描画
      size: 15,
      height: 6
    }),
    // マテリアルの設定(スムースシェーディングを指定)
    new THREE.MeshPhongMaterial({ color: 0xffffff, shading: THREE.SmoothShading })
```

```
  );
  // テキストの回転角度と位置を設定
  text3D.rotation.y = Math.PI / 3;
  text3D.position.x = -30;
  // シーンを初期化する
  var scene = new THREE.Scene();
  // テキストを3Dシーンに追加
  scene.add(text3D);
  // カメラを3Dシーンに追加
  scene.add(camera);
  // 光源を追加
  var light = new THREE.DirectionalLight(0x0000ff, 2);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  // 霧(フォグ)を追加
  // scene.fog = new THREE.Fog(0xffffff, 50, 220);
  scene.fog = new THREE.FogExp2(0xffffff, 0.005);
  // 霧(フォグ)を物体に適用
  text3D.fog = true;
  // レンダリング
  renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、WebGL&JavaScriptの文字に霧(フォグ)が適用されます。



ONEPOINT 霧（フォグ）を設定するには「fog」プロパティを使う

Three.jsで霧（フォグ）を設定するには、シーンの「fog」プロパティにフォグオブジェクトを設定します。霧（フォグ）を設定できるオブジェクト（コンストラクタ）は、「THREE.Fog()」と「THREE.FogExp2()」があります。「THREE.Fog()」のパラメータは、霧の色と霧の手前の距離、奥までの距離の3つのパラメータを指定します。距離はデフォルトでは手前の距離が「1」、奥までの距離が「1000」になっています。「THREE.FogExp2()」のパラメータは、霧の色と霧の濃さです。デフォルトでは霧の濃さは「0.00025」になっています。

シーンの「fog」プロパティに霧を処理するオブジェクトを設定しただけでは、物体に霧は適用されません。物体に霧が適用されるようにするには、物体の「fog」プロパティに「true」を設定します。

SECTION-250

立体の影を表示する

ここでは、WebGLで手軽に3D処理が行えるThree.jsライブラリを使って影を表示する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

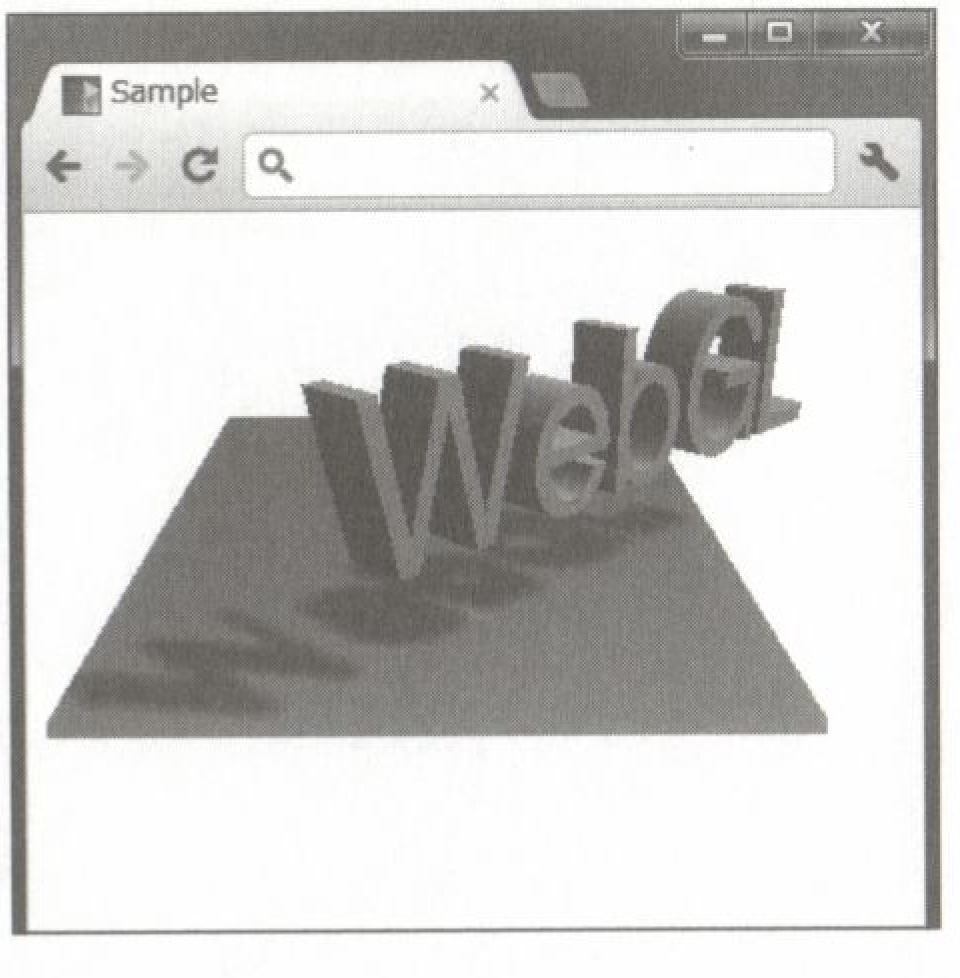
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/helvetiker_regular.typeface.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  var renderer = new THREE.WebGLRenderer({antialias: true});
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(0, 50, 100);
  camera.lookAt( {x:0, y:0, z:0 } );
  // テキストを初期化する
  var text3D = new THREE.Mesh(
    // 文字データの設定
    new THREE.TextGeometry(
      // 描画する文字を設定
      "WebGL", {
      // サイズと高さを描画
      size: 15,
      height: 6
    }),
    // マテリアルの設定(スムースシェーディングを指定)
    new THREE.MeshPhongMaterial({ color: 0xffffff, shading: THREE.SmoothShading })
  );
```

```
  // テキストの回転角度と位置を設定
  text3D.rotation.y = Math.PI / 4;
  text3D.position.set(-10, 15, 20);
  // 平面を作成
  var plane = new THREE.Mesh(
    new THREE.PlaneGeometry(70, 100),
    new THREE.MeshLambertMaterial({color: 0x70ff70})
  );
  // 平面の位置を設定
  plane.position.set(0,0, -30);
  // シーンを初期化する
  var scene = new THREE.Scene();
  // テキストを3Dシーンに追加
  scene.add(text3D);
  // 平面を3Dシーンに追加
  scene.add(plane);
  // カメラを3Dシーンに追加
  scene.add(camera);
  // 光源を追加
  var light = new THREE.SpotLight(0xffffff, 1, 0, true);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  // 影が表示されるようにレンダラーや物体に設定
  renderer.shadowMapEnabled = true;
  light.castShadow = true;
  text3D.castShadow = true;
  plane.receiveShadow = true;
  // レンダリング
  renderer.render(scene, camera);
}, false);
```

ブラウザでindex.htmlを表示すると、WebGLの文字の影が文字よりも下にある平面に表示されます。



ONEPOINT 立体の影を表示するには「shadowMapEnabled」プロパティなどを使う

Three.jsで影を表示するには、いくつかの条件と設定が必要です。まず、物体に影を落とすことができる光源はスポットライトのみです。環境光や点光源では、影は描画できません。また、レンダラーの「shadowMapEnabled」プロパティに「true」を設定する必要があります。

実際に影を落とす物体には「castShadow」プロパティに「true」を設定し、影を受ける物体には「receiveShadow」プロパティを「true」に設定します。影を落としつつ、さらに他の物体の影も受ける場合は、「castShadow」プロパティと「receiveShadow」プロパティの両方に「true」を設定する必要があります。

なお、Safari 5.xでは、「receiveShadow」プロパティに「true」を設定した物体があるとページが延々とリロードされ、最後には表示されなくなってしまう不具合が発生することがあります。

関連項目 ▶▶▶

● 光源を追加する ……………………………………………………… p.542

SECTION-251

WebGLが使えないデバイスでも3D描画を可能にする

ここでは、iPhone/iPadやAndroidなどWebGLがブラウザから使えないデバイスでも3D描画を可能にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

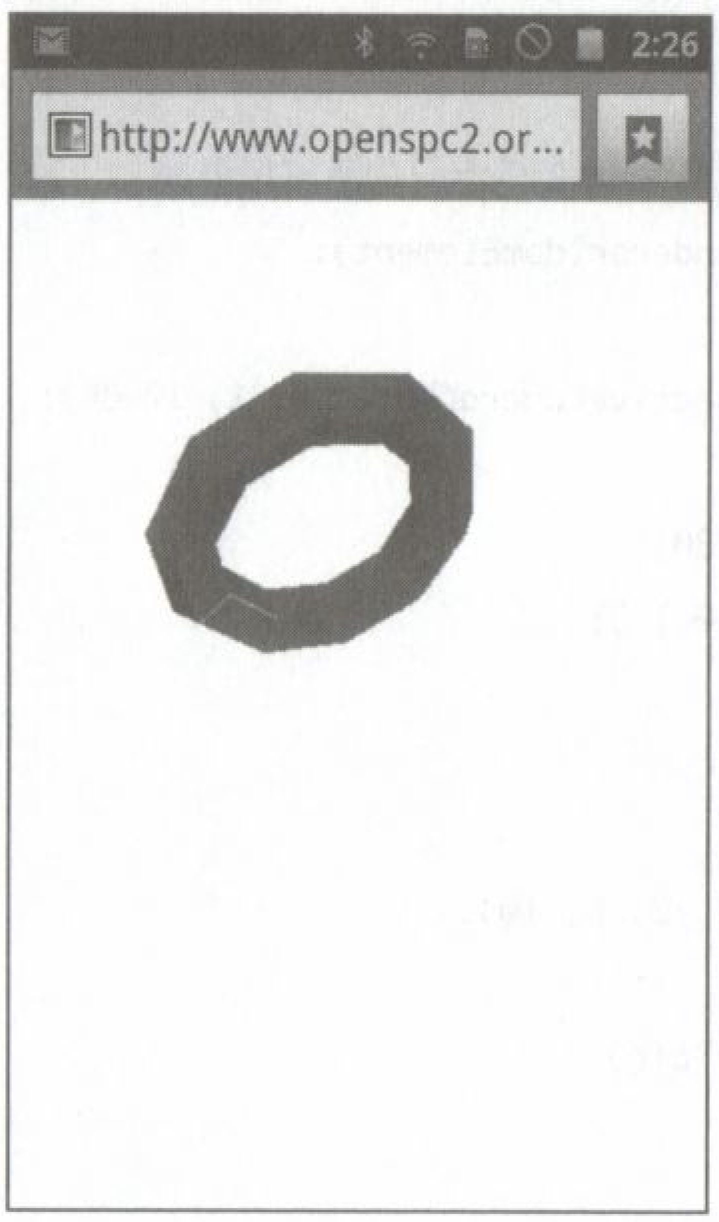
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/Three.js"></script>
    <script src="js/sample.js"></script>
  </head>
  <body></body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Three.jsを初期化
  // var renderer = new THREE.WebGLRenderer();
  var renderer = new THREE.CanvasRenderer();
  // 描画領域のサイズは縦横とも320ピクセル
  renderer.setSize(320, 320);
  // Three.jsによって生成されたCanvasをページの末尾に追加
  document.body.appendChild(renderer.domElement);
  // カメラを初期化
  var camera = new THREE.PerspectiveCamera(40, 1.0, 1, 10000);
  // カメラ位置を設定
  camera.position.set(50, 40, 30);
  camera.lookAt( {x:0, y:0, z:0 } );
  // オブジェクトを初期化する
  var torus = new THREE.Mesh(
    // トーラスのサイズを設定
    new THREE.TorusGeometry(12, 3, 5, 10),
    // マテリアルをLambertに
    new THREE.MeshLambertMaterial()
  );
  // シーンを初期化する
  var scene = new THREE.Scene();
  // トーラスを3Dシーンに追加
  scene.add(torus);
  // カメラを3Dシーンに追加
  scene.add(camera);
```

```
  // 光源を追加
  var light = new THREE.DirectionalLight(0x0000ff, 1.25);
  // 光源の位置を設定
  light.position.set(200, 260, 100);
  scene.add(light);
  // 回転角度を入れる変数を用意
  var rotX = 0;
  var rotY = 0;
  // レンダリング。アニメーション毎に呼び出す
  function render(){
    // トーラスのXY軸を回転
    rotX = rotX + 0.02;
    rotY = rotY + 0.01;
    torus.rotation.x = rotX;
    torus.rotation.y = rotY;
    // 定期的に描画するようにする
    requestAnimationFrame(render);
    renderer.render(scene, camera);
  }
  render();
}, false);
```

WebGL表示ができないAndroid標準ブラウザやiPhone/iPadのブラウザでも3D表示が可能になります。



ONEPOINT **iPhoneやAndroidで3D描画を可能にするには「new THREE.CanvasRenderer()」とする**

iPhoneやAndroidのWebブラウザは、WebGLに対応していません。このため、WebGLを利用した3D描画を行うことはできません。ただし、Three.jsには、WebGLが使えない環境でもCanvasに3D描画を行うことで対処する方法があります。

Canvasに描画する場合は、「THREE.WebGLRenderer()」の代わりに、「THREE.CanvasRenderer()」のレンダラーを指定し、「new THREE.CanvasRenderer()」とします。ただし、すべてが同じように表示されるわけではありません。また、2DのCanvasに描画しているため、処理速度がかなり低下することもあります(ポリゴン数を減らせば処理速度は向上する)。

COLUMN **IE9で表示されない場合について**

IE9の場合は、F12キーを押して開発者ツールを表示しておかないと、サンプルは動作しません。これは、IEでは開発者ツールを表示し、ページをリロードしないと、「console」オブジェクトにアクセスできないためです。

同様の理由で、IEを搭載しているWindows Phone(7.5)では開発者ツールがないため、動作しません。これを解消するには、THREE.jsライブラリの先頭(「'use strict'」の次)に、次のコードを追加します。これで、Windows Phoneでも動作するようになります。

```
console={};console.warn=console.error=console.log=function(){};
```

関連項目 ▶▶▶

● WebGLが利用できるか調べる ……… p.532

SECTION-252

SVGの図形のサイズを変更する

ここでは、インラインSVGを使って表示された図形のサイズを変更する方法について解説します。

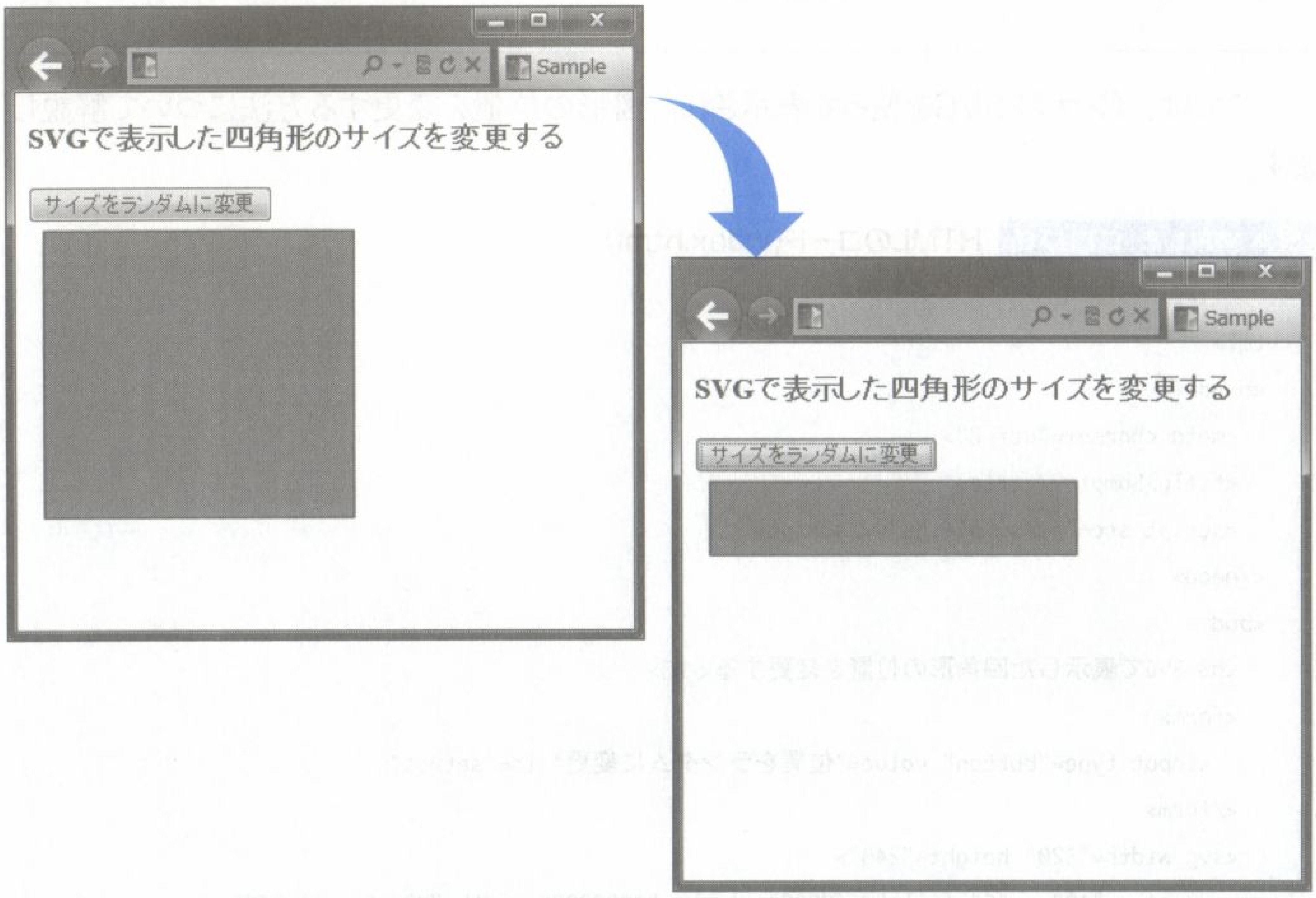
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>SVGで表示した四角形のサイズを変更する</h3>
    <form>
      <input type="button" value="サイズをランダムに変更" id="check">
    </form>
    <svg width="320" height="240">
      <rect x="10" y="5" fill="#ff0000" stroke="#000000" width="200" height="180" />
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("check").addEventListener("click", function(){
    // 乱数値を算出
    var w = Math.floor(Math.random()*100)*3+10;
    var h = Math.floor(Math.random()*100)*2+10;
    // SVGの最初のrect要素を取得
    var svgRect = document.getElementsByTagName("rect")[0];
    // 四角形の横幅と縦幅を設定
    svgRect.setAttribute("width", w);
    svgRect.setAttribute("height", h);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、SVGで描画された四角形のサイズがランダムに変わります。



ONEPOINT

SVGの図形のサイズを変更するには図形を示す要素の「width」属性と「height」属性の値を変更する

インラインSVGを使って描画されたSVGの図形のサイズを変更するには、図形を示す要素の「width」属性と「height」属性の値を変更します。属性値を変更するには、「setAttribute()」メソッドを使います。「setAttribute()」メソッドの最初のパラメータには変更したい属性名を文字列で指定し、2番目のパラメータには設定するサイズを指定します。

関連項目 ▶▶▶

- SVGの図形の位置を変更する ……………………………………………… p.594
- SVGの図形を変形する ………………………………………………………… p.596

SECTION-253

SVGの図形の位置を変更する

ここでは、インラインSVGを使って表示された図形の位置を変更する方法について解説します。

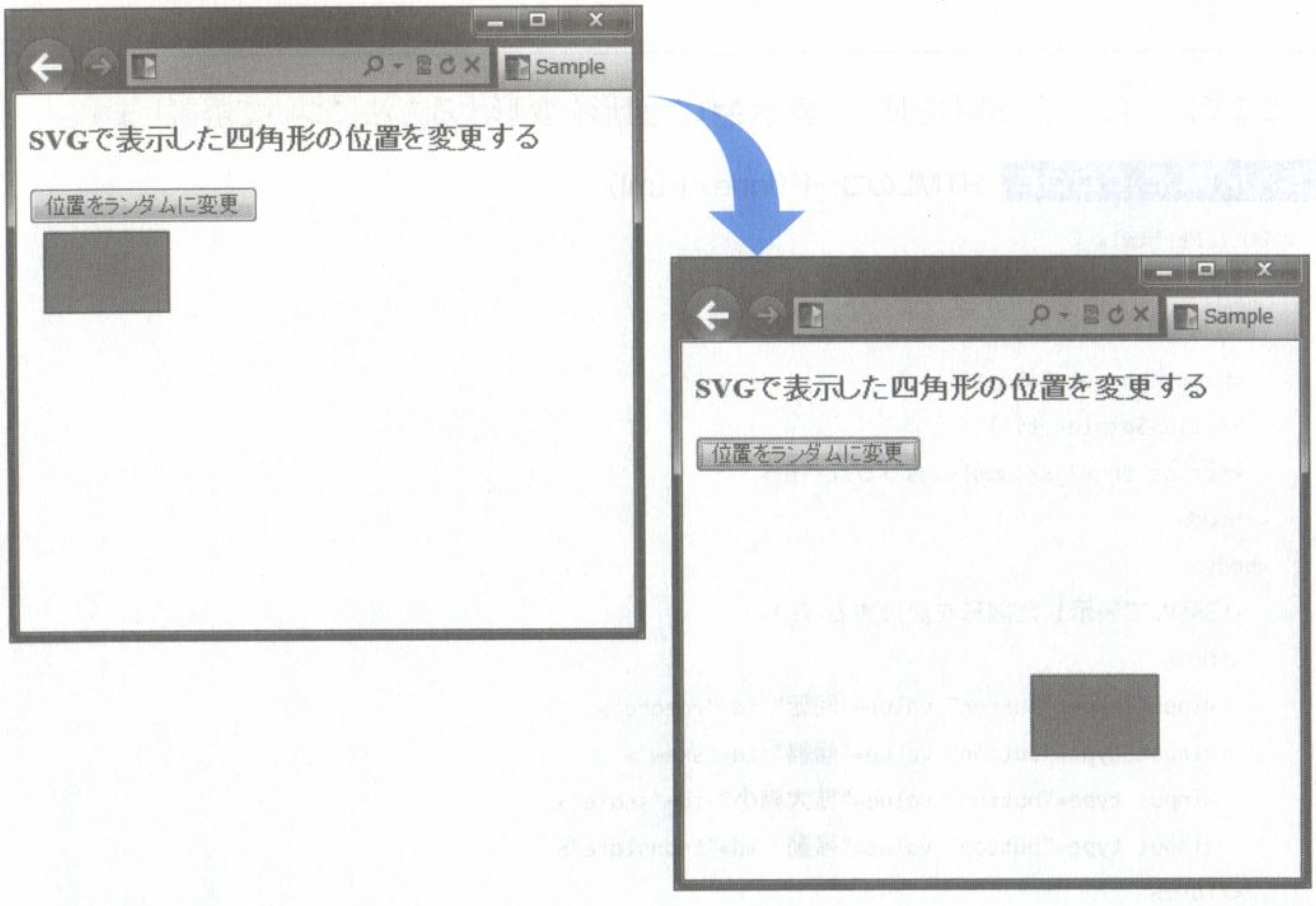
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>SVGで表示した四角形の位置を変更する</h3>
    <form>
      <input type="button" value="位置をランダムに変更" id="setPos">
    </form>
    <svg width="320" height="240">
      <rect x="10" y="5" fill="#ff0000" stroke="#000000" width="80" height="50" />
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setPos").addEventListener("click", function(){
    // 乱数値を算出
    var newX = Math.floor(Math.random()*90)*3+10;
    var newY = Math.floor(Math.random()*70)*2+10;
    // SVGの最初のrect要素を取得
    var svgRect = document.getElementsByTagName("rect")[0];
    // 四角形のX座標とY座標を設定
    svgRect.setAttribute("x", newX);
    svgRect.setAttribute("y", newY);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、SVGで描画された四角形の位置がランダムに変わります。



ONEPOINT

SVGの図形の位置を変更するには 図形を示す要素の「x」属性と「y」属性の値を変更する

インラインSVGを使って描画されたSVGの図形の位置を変更するには、図形を示す要素の「x」属性と「y」属性の値を変更します。属性値を変更するには、「setAttribute()」メソッドを使います。「setAttribute()」メソッドの最初のパラメータには変更したい属性名を文字列で指定し、2番目のパラメータには設定する位置を指定します。

関連項目 ▶▶▶

- SVGの図形のサイズを変更する …… p.592
- SVGの図形を変形する …… p.596

SVGの図形を変形する

ここでは、インラインSVGを使って表示された図形を変形する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

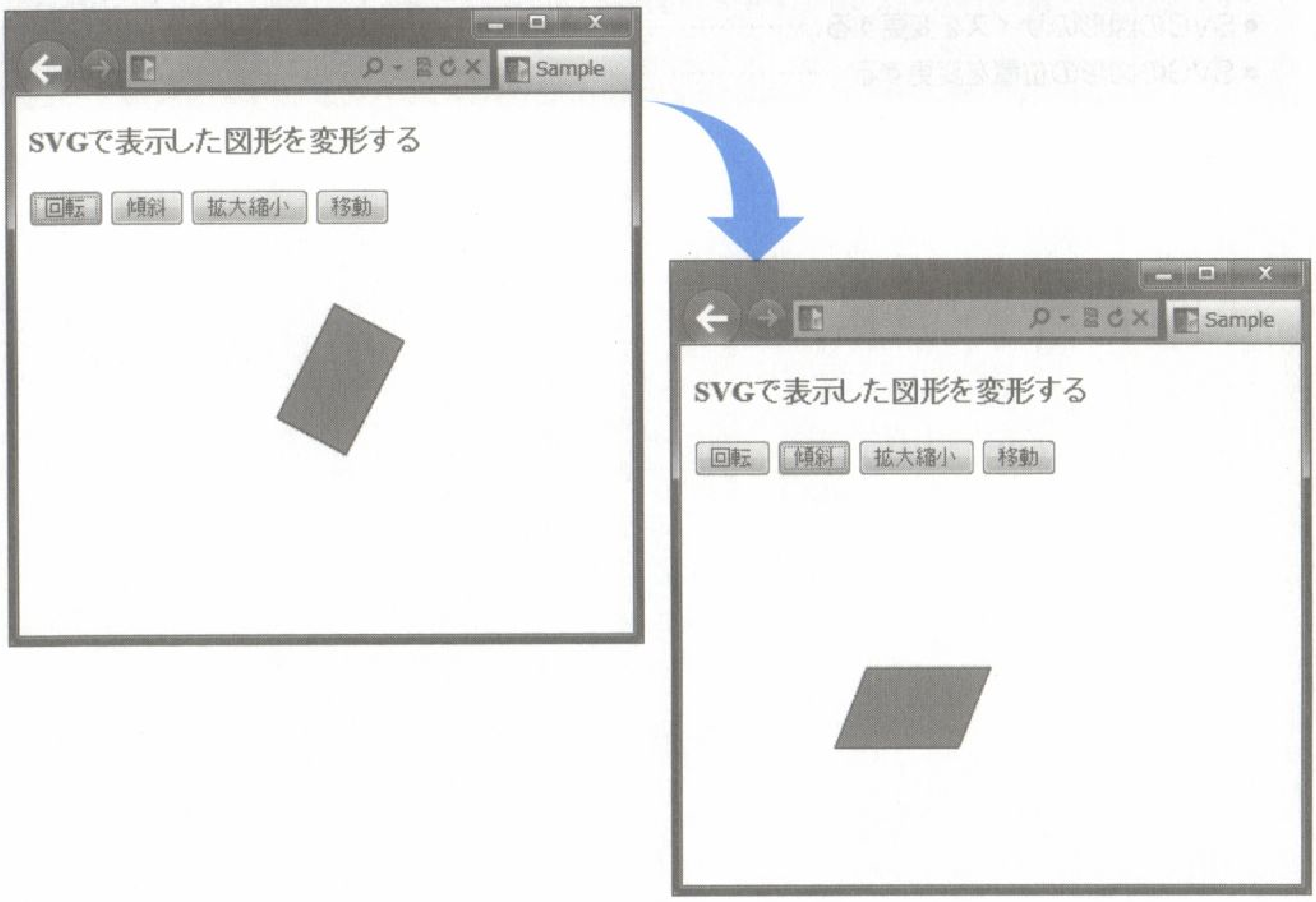
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>SVGで表示した図形を変形する</h3>
    <form>
      <input type="button" value="回転" id="rotate">
      <input type="button" value="傾斜" id="skew">
      <input type="button" value="拡大縮小" id="scale">
      <input type="button" value="移動" id="translate">
    </form>
    <svg width="320" height="240">
      <rect x="160" y="120" fill="#ff0000" stroke="#000000" width="80" height="50" />
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // SVGの最初のrect要素を取得
  var svgRect = document.getElementsByTagName("rect")[0];
  // ボタンにイベントを設定【回転】
  document.getElementById("rotate").addEventListener("click", function(){
    // 乱数値を算出
    var r = Math.random()*360;
    // transform属性に値を設定。2番目と3番目は回転の中心
    svgRect.setAttribute("transform", "rotate("+r+", 160, 120)");
  }, false);
  // ボタンにイベントを設定【傾斜】
  document.getElementById("skew").addEventListener("click", function(){
    // 乱数値を算出
    var sr = Math.random()*60 - 30;
    // transform属性に値を設定
    svgRect.setAttribute("transform", "skewX("+sr+")");
  }, false);
```

```
// ボタンにイベントを設定【拡大縮小】
document.getElementById("scale").addEventListener("click", function(){
  // 乱数値を算出
  var s = Math.random()*2-0.5;
  // transform属性に値を設定
  svgRect.setAttribute("transform", "scale("+s+","+s+")");
}, false);
// ボタンにイベントを設定【移動】
document.getElementById("translate").addEventListener("click", function(){
  // 乱数値を算出
  var x = Math.random()*280 - 120;
  var y = Math.random()*120 - 60;
  // transform属性に値を設定
  svgRect.setAttribute("transform", "translate("+x+","+y+")");
 }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、ボタンに対応した変形処理が行われます。



ONEPOINT SVGの図形を変形するには「transform」属性を使う

インラインSVGを使って描画されたSVGを変形するには、「transform」属性に変形を示す文字列を指定します。変形させるには次に示す文字列と値をセットにします。「setAttribute()」メソッドの最初のパラメータには「"transform"」を文字列で指定し、2番目のパラメータには変形内容を示す文字列を指定します。

命令	パラメータ	処理内容
matrix	a b c d e f	行列を利用して変形
rotate	角度 回転の中心XY座標	回転。回転の中心座標は省略できる
scale	横の倍率 縦の倍率	拡大縮小(スケール)
skewX	角度	横方向の傾斜
skewY	角度	縦方向の傾斜
translate	X座標 Y座標	移動

複数の変形をまとめて行うには、「matrix」を使うか、「scale(2) skewX(-2) rotate(45)」のように半角空白で区切って列記します。

関連項目 ▶▶▶

- SVGの図形のサイズを変更する ………… p.592
- SVGの図形の位置を変更する ………… p.594

SECTION-255

SVGの図形の塗りや線の色を変更する

ここでは、インラインSVGを使って表示された図形の塗りや線の色を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

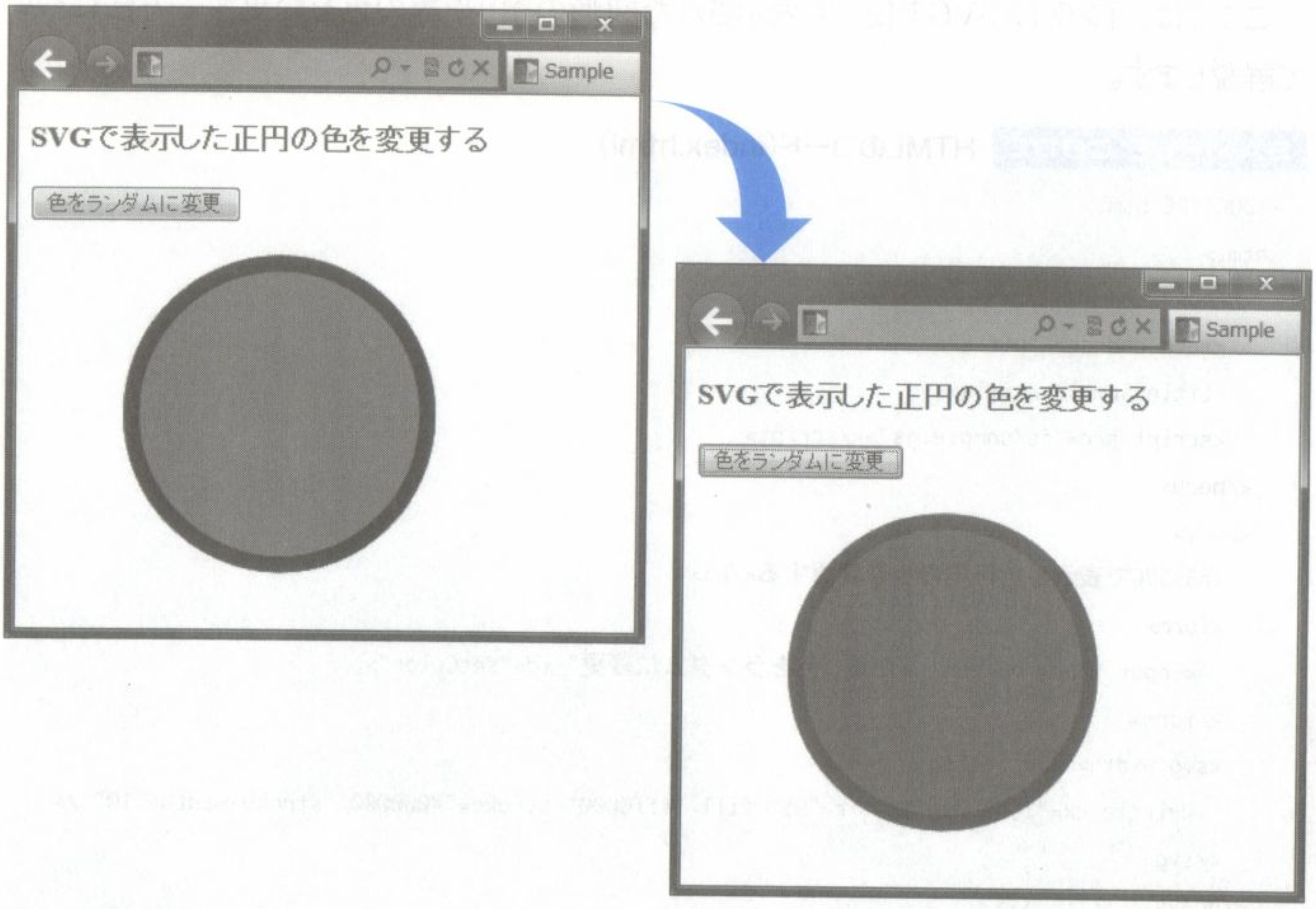
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>SVGで表示した正円の色を変更する</h3>
    <form>
      <input type="button" value="色をランダムに変更" id="setColor">
    </form>
    <svg width="320" height="240">
      <circle cx="160" cy="120" r="95" fill="#ff0000" stroke="#000000" stroke-width="10" />
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setColor").addEventListener("click", function(){
    // 乱数値を算出(IE9/Firefoxは#rrggbb形式でないと駄目)
    var fR = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var fG = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var fB = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var sR = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var sG = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var sB = ("0"+Math.floor(Math.random() * 256)).slice(-2);
    var color1 = "#"+fR+fG+fB;
    var color2 = "#"+sR+sG+sB;
    // SVGの最初のrect要素を取得
    var svgRect = document.getElementsByTagName("circle")[0];
    // 塗りの色を設定
    svgRect.setAttribute("fill", color1);
    // 線の色を設定
    svgRect.setAttribute("stroke", color2);
  }, false);
```

```
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、SVGで描画された円の塗りや線の色がランダムに変わります。



ONEPOINT **SVGの図形の塗りや線の色を変更するには図形を示す要素の「fill」属性や「stroke」属性の値を変更する**

インラインSVGを使って描画されたSVGの塗りの色を変更するには、図形を示す要素の「fill」属性の値を変更します。線の色を変更するには、「stroke」属性の値を変更します。これらの属性値を変更するには、「setAttribute()」メソッドを使います。「setAttribute()」メソッドの最初のパラメータには変更したい属性名を文字列で指定し、2番目のパラメータには設定する位置を指定します。

色を設定する場合、IE9とFirefoxでは、「#rrggbb」形式の文字列にしないと、色が反映されません。

SECTION-256

SVGのテキストの内容を変更する

ここでは、インラインSVGを使って表示されたテキストの内容を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>SVGで表示された文字を変更する</h3>
    <svg width="320" height="240">
      <text id="clock" x="50" y="100" font-size="60">12:34:56</text>
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  setInterval(function(){
  // 現在の日付情報を取得(生成)
  var dateObj = new Date();
  var h = ("0"+dateObj.getHours()).slice(-2);
  var m = ("0"+dateObj.getMinutes()).slice(-2);
  var s = ("0"+dateObj.getSeconds()).slice(-2);
  // SVGテキスト要素を取得
  var svgText = document.getElementById("clock");
  // 文字を変更
  svgText.textContent = h+":"+m+":"+s;
  }, 1000);
}, false);
```

ブラウザでindex.htmlを表示すると、SVGのテキストの内容がリアルタイムに変更されます。



ONEPOINT　**SVGのテキストの内容を変更するには「textContent」プロパティを使う**

インラインSVGを使って描画されたSVGのテキストの内容を変更するには、「textContent」プロパティの値を変更します。また、
や改行コード(10、13/LF、CR)が含まれていても、その部分で改行されることはありません。

また、文字サイズなど文字関連の属性を変更する場合には、「svgText.setAttribute("font-size", 9);」のように、「setAttribute()」メソッドを使って行います。

SECTION-257

SVGの図形のグラデーションの種類を変更する

ここでは、インラインSVGを使って表示された図形のグラデーションの種類を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

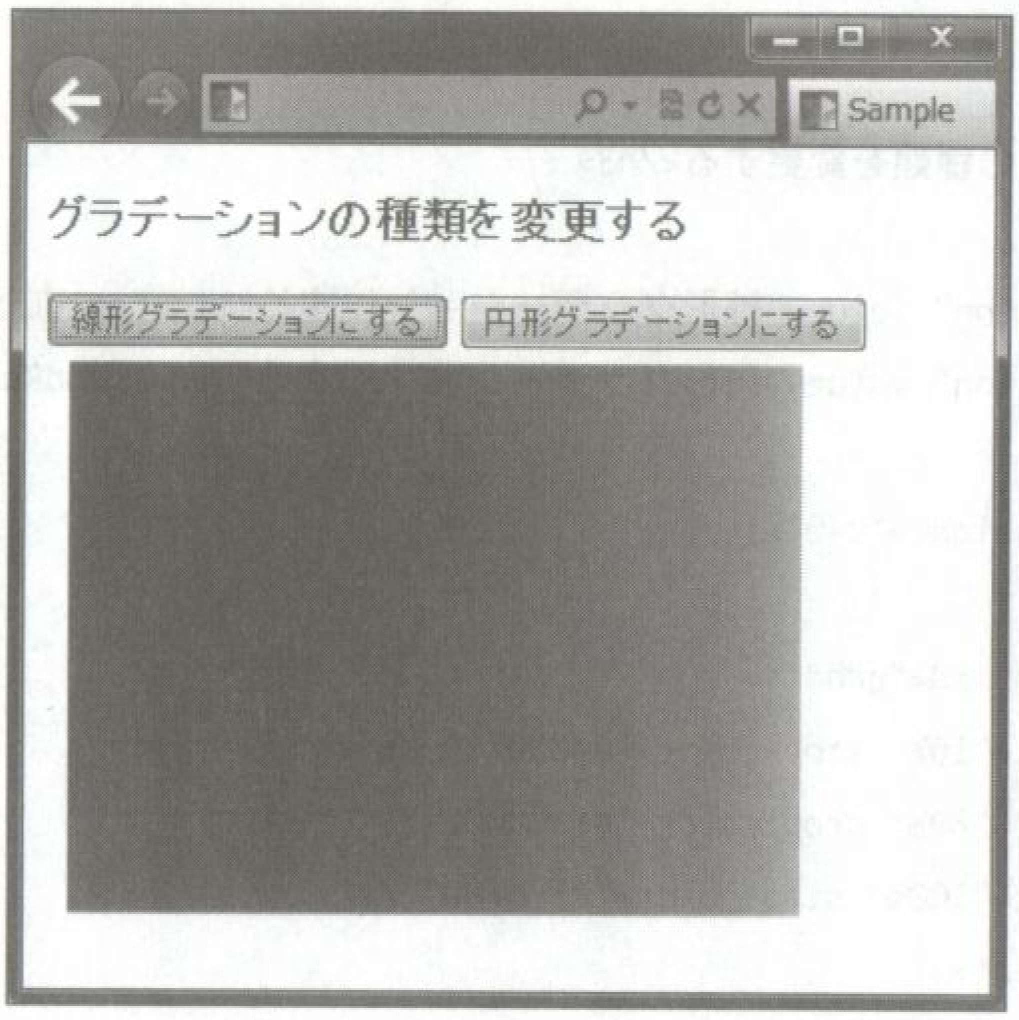
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>グラデーションの種類を変更する</h3>
    <form>
      <input type="button" value="線形グラデーションにする" id="gradLine">
      <input type="button" value="円形グラデーションにする" id="gradRadial">
    </form>
    <svg width="320" height="240">
      <defs>
        <linearGradient id="grad1">
          <stop offset="10%" stop-color="#0000ff" />
          <stop offset="80%" stop-color="#ff0000" />
          <stop offset="100%" stop-color="#ffbf00" />
        </linearGradient>
        <radialGradient id="grad2">
          <stop offset="10%" stop-color="#0000ff" />
          <stop offset="80%" stop-color="#ff0000" />
          <stop offset="100%" stop-color="#ffbf00" />
        </radialGradient>
      </defs>
      <rect x="10" y="5" fill="url(#grad1)" width="300" height="220" />
    </svg>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

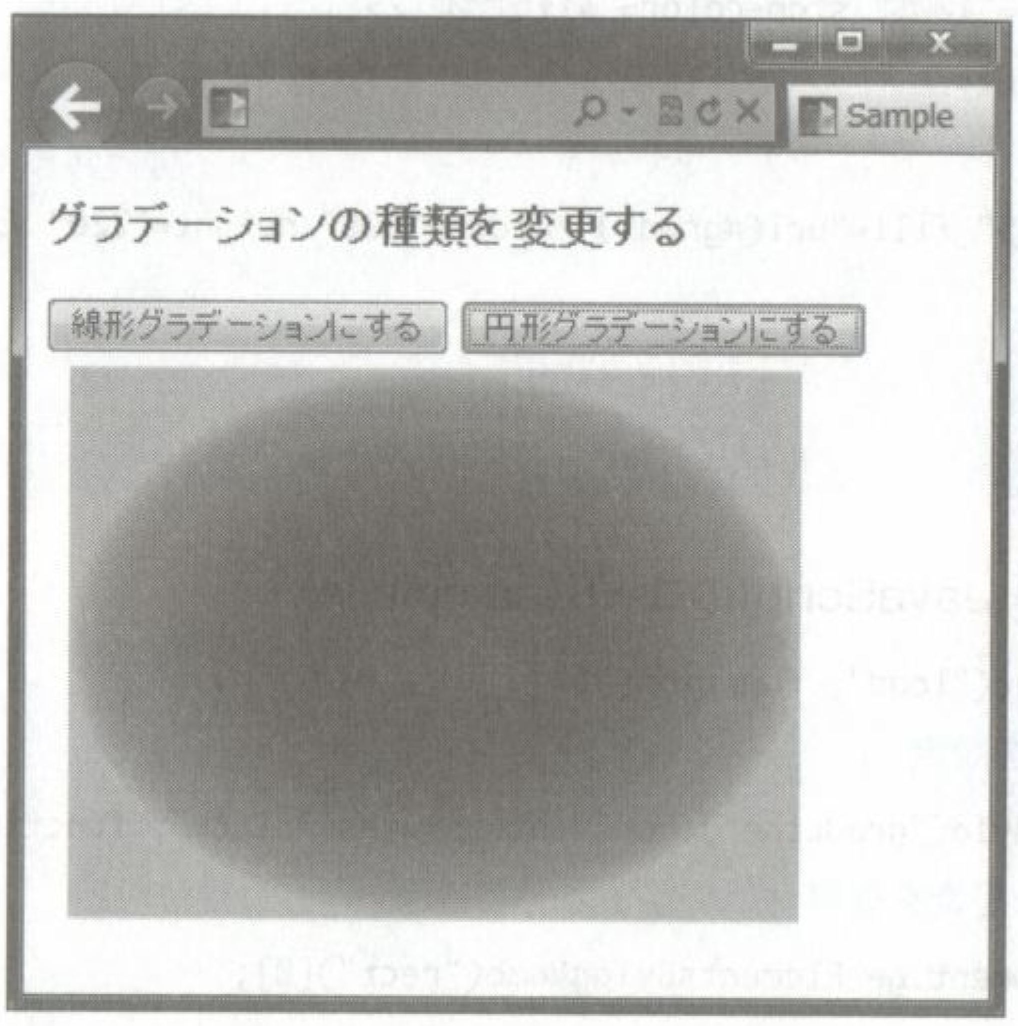
```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("gradLine").addEventListener("click", function(){
    // SVGの最初のrect要素を取得
    var svgRect = document.getElementsByTagName("rect")[0];
```

```
      // 塗りに線形グラデーションを設定
      svgRect.setAttribute("fill", "url(#grad1)");
    }, false);
    // ボタンにイベントを設定
    document.getElementById("gradRadial").addEventListener("click", function(){
      // SVGの最初のrect要素を取得
      var svgRect = document.getElementsByTagName("rect")[0];
      // 塗りに円形グラデーションを設定
      svgRect.setAttribute("fill", "url(#grad2)");
    }, false);
  }, false);
```

ブラウザでindex.htmlを表示し、線形グラデーションにするボタンをクリックすると、四角形の塗りが直線的なグラデーションで表示されます。



円形グラデーションにするボタンをクリックすると、四角形の塗りが円状のグラデーションで表示されます。



ONEPOINT **SVGの図形のグラデーションの種類を変更するには 図形を示す要素の「fill」属性の値を変更する**

インラインSVGを使って描画された図形のグラデーションの種類を変更するには、図形を示す要素の「fill」属性の値を変更します。SVGではグラデーションも塗りの1つとして処理されます。そこで、「setAttribute()」メソッドを使って表示したいグラデーションを示す文字列である「"url(#定義したグラデーション名)"」を「fill」属性に設定します。

CHAPTER 11

Audio/Video/HTML Media

オーディオオブジェクトを生成する

ここでは、オーディオオブジェクトを生成する方法について解説します。

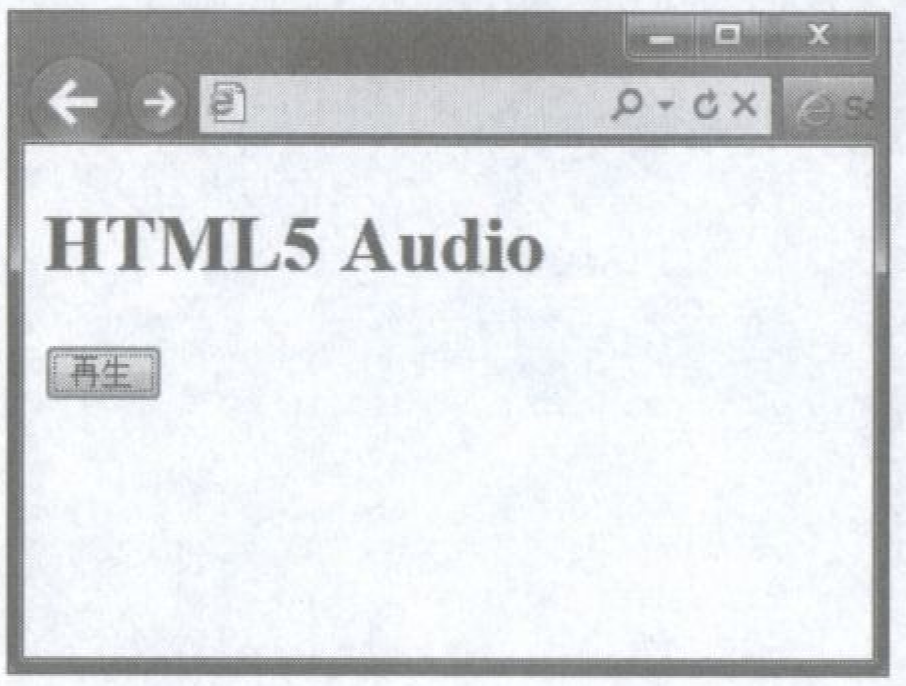
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <form>
      <input type="button" value="再生">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(){
    // オーディオオブジェクトの生成とオーディオファイルの読み込み
    var audioObj = new Audio("music/sample.mp3");
    // オーディオを再生する
    audioObj.play();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、オーディオファイルが読み込まれて再生が行われます。



ONEPOINT オーディオオブジェクトを生成するには「new Audio()」を使う

HTML5でオーディオを扱うには、2つの方法があります。1つは、HTML文書内に書かれた「audio」要素にアクセスし、制御する方法です。もう1つが「new Audio()」を使って「Audio」オブジェクトを生成し、制御する方法です。前者はDOMの扱いになるので、「document.getElementById()」などを使ってアクセスすることができます（356ページ参照）。「new Audio()」の場合は、パラメータを指定しなければオーディオオブジェクトが生成され、URLを指定した場合はサウンドファイルとして読み込まれます。

オーディオオブジェクトは「audio」要素でも「new Audio()」で生成しても同じなので、オーディオ処理に使用できるメソッドとプロパティに違いはありません。

COLUMN 音を生成する他の方法

HTML5では、他にも音を生成する「Web Audio API」（Google Chrome/Safari 6）、「Audio Data API」（Firefox）もあります。詳細については、それぞれ、次のWebページを参照してください。なお、Web Audio APIについて、848〜856ページで解説しています。

- Web Audio API
  URL https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html
- Audio Data API
  URL https://wiki.mozilla.org/Audio_Data_API

COLUMN FirefoxやOperaでの音声の再生について

FirefoxやOperaではOgg形式でないと、オーディオが再生されません。本書のサンプルでは、MP3形式の音声ファイルを使っているため、そのままではオーディオが再生されず、動作の確認を行うことができません。FirefoxやOperaで動作確認を行う際は、別途、Ogg形式のオーディオを用意し、サンプルのコードを適宜、書き換えてください。

関連項目 ▶▶▶

● Web Audioで音を鳴らす …… p.850

SECTION-259

オーディオを再生・停止する

ここでは、オーディオを再生・停止する方法について解説します。

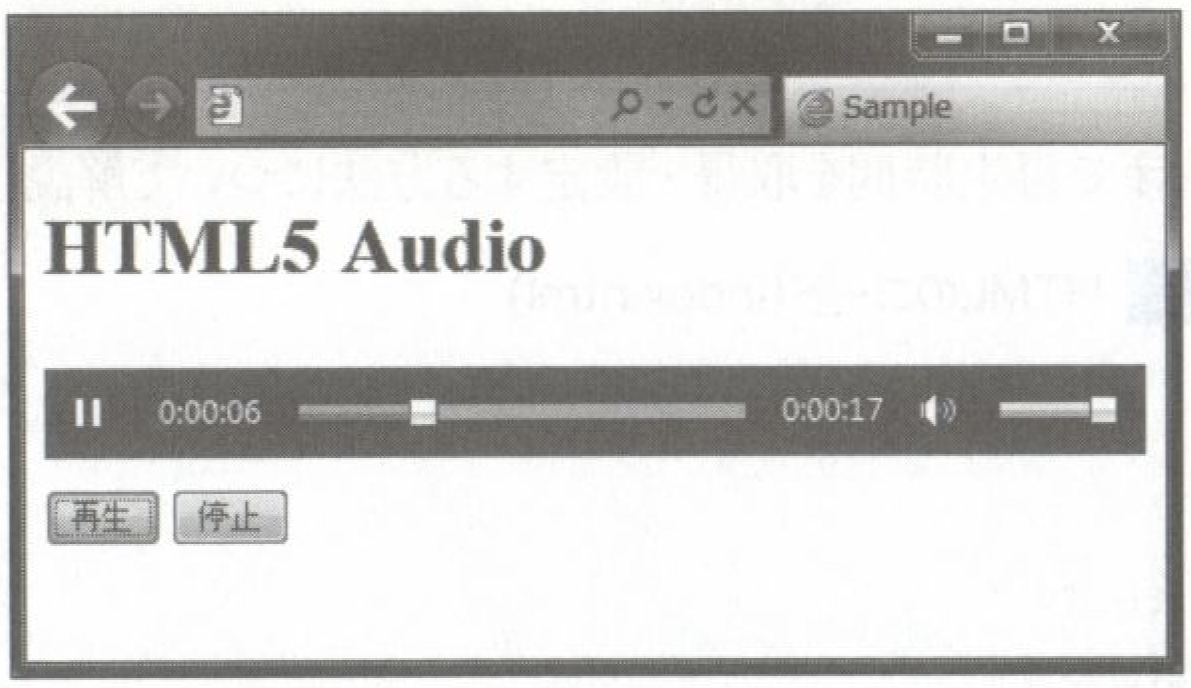
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="button" value="再生">
      <input type="button" value="停止">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オーディオ要素を読み出す
  var audioObj = document.getElementsByTagName("audio")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  btn[0].addEventListener("click", function(){
    // オーディオを再生
    audioObj.play();
  }, false);
  btn[1].addEventListener("click", function(){
    // オーディオの再生を停止
    audioObj.pause();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、再生ボタンをクリックすると、オーディオファイルが読み込まれて再生されます。停止ボタンをクリックすると、再生が停止します。



ONEPOINT

オーディオを再生するには「play()」メソッド、停止するには「pause()」メソッドを使う

HTML5 Audioを再生するには、オーディオオブジェクトの「play()」メソッドを使います。「play()」メソッド実行時にオーディオファイルが読み込まれていない場合でもエラーにはならず、再生可能になった段階で再生が行われます。

オーディオを停止するには、「pause()」メソッドを使います。「pause()」メソッドは、再生を停止するだけで巻き戻し処理などは行いません。このため、「pause()」メソッド実行後に再度、「play()」メソッドを実行すると、停止したところから再生が開始されます。

関連項目 ▶▶▶

- オーディオの再生時のイベントを取得する …… p.631
- オーディオの再生が終了した際のイベントを取得する …… p.633
- Web Audioで音を鳴らす …… p.850
- Web Audioでファイルを読み込んで演奏する …… p.852

SECTION-260

再生時間を取得・設定する

ここでは、オーディオで再生時間を取得・設定する方法について解説します。

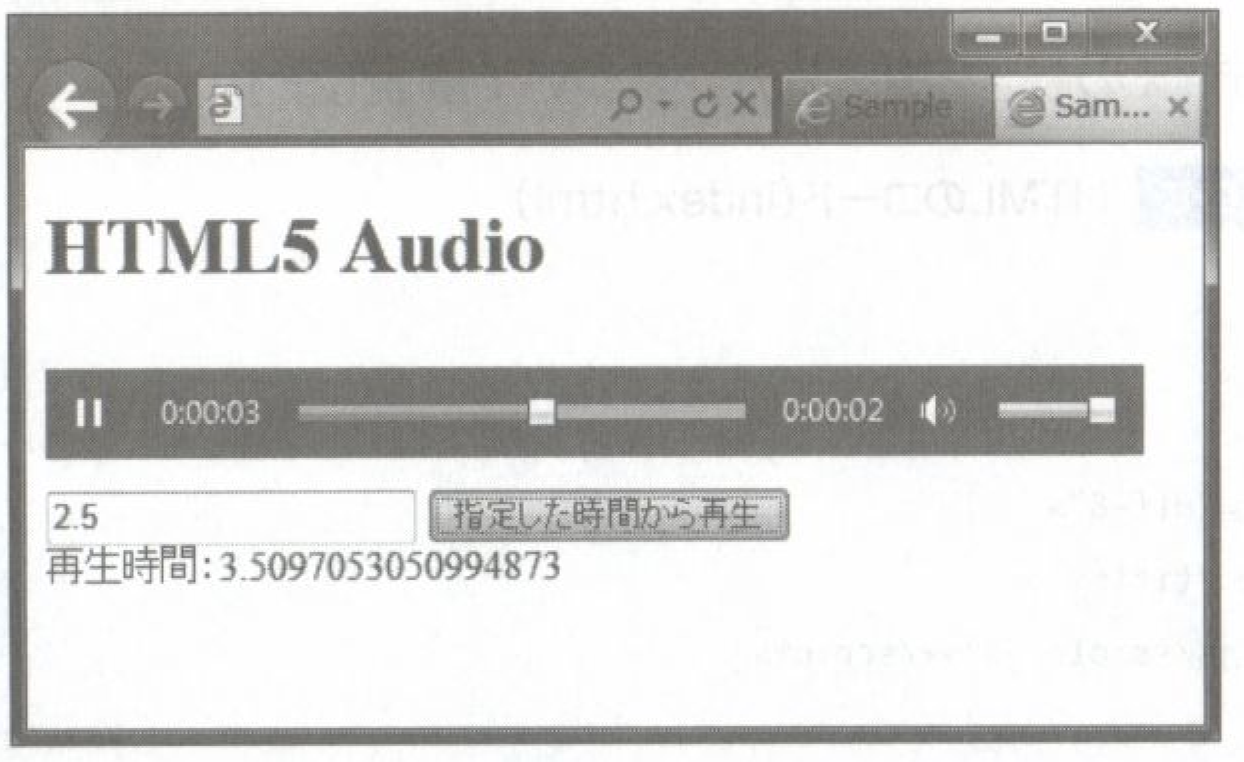
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="text" value="2.5" id="startTime">
      <input type="button" value="指定した時間から再生">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オーディオ要素を読み出す
  var audioObj = document.getElementsByTagName("audio")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  btn[1].addEventListener("click", function(){
    // 設定時間を読み出し
    var time = document.getElementById("startTime").value;
    // 時間を設定
    audioObj.currentTime = time;
    // オーディオを再生
    audioObj.play();
  }, false);
  // オーディオにイベントを割り当て
  audioObj.addEventListener("timeupdate", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生時間を表示
    ele.innerHTML = "再生時間 :"+this.currentTime;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、開始時間を入力後、再生ボタンをクリックすると、オーディオファイルが指定した時間から再生されます。また、再生中には、画面に再生時間が表示されます。



ONEPOINT

演奏開始時間と再生時間を取得・設定するには「currentTime」プロパティを使う

現在の再生時間を取得するには、オーディオオブジェクトの「currentTime」プロパティを参照します。「currentTime」プロパティの内容は再生されている時間であり、「5.963」のようにミリ秒を含む値になります。

指定した時間から再生する場合には「currentTime」プロパティに、再生を開始したい時間を設定します。なお、再生中でも、停止中でも、「currentTime」プロパティに値を設定することができます。

Windows Phone(7.5)ではオーディオの再生がユーザーによって行われていない場合、「currentTime」プロパティの値を読み出すことはできますが、設定することはできません。一度、オーディオを再生した後であれば、「currentTime」プロパティに値を設定し、任意の秒数から再生できるようになります。

SECTION-261

オーディオの長さを調べる

ここでは、オーディオの長さを調べる方法について解説します。

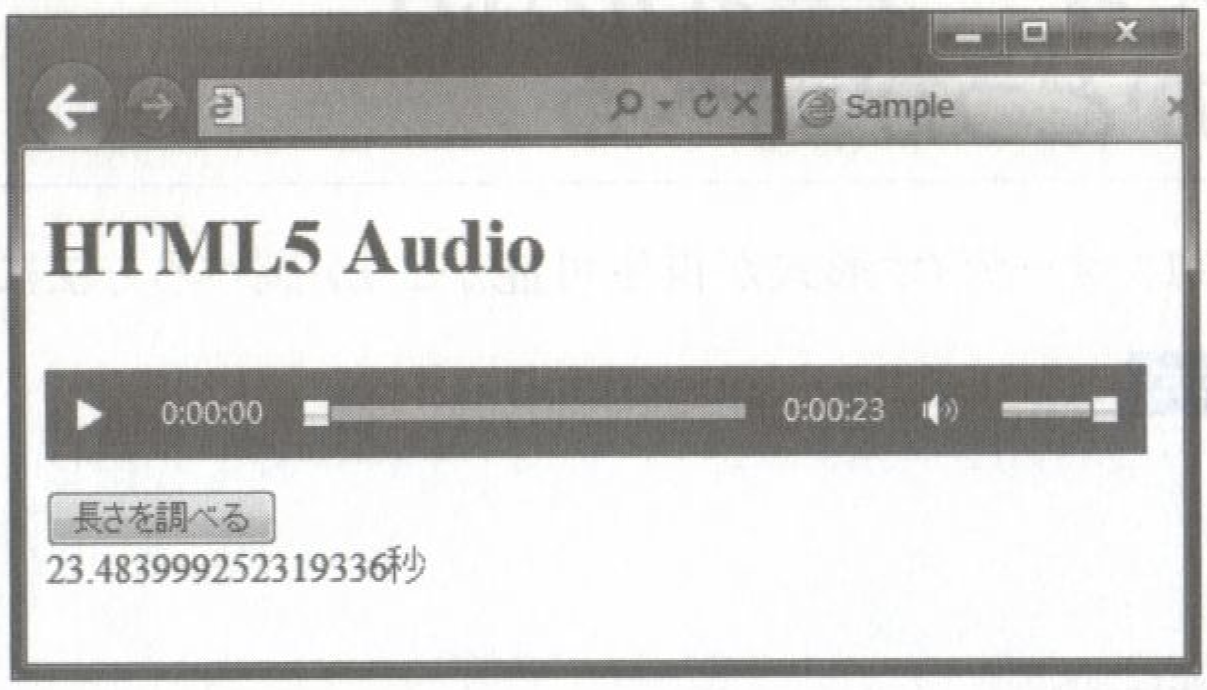
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="button" value="長さを調べる">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input")[0];
  // オーディオオブジェクトを生成
  var audioObj = document.getElementsByTagName("audio")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(){
    // オーディオファイルの長さ(尺)を変数に代入
    var sec = audioObj.duration;
    document.getElementsByTagName("output")[0].innerHTML = sec+"秒";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、オーディオの長さが表示されます。



ONEPOINT　オーディオの長さを取得するには「duration」プロパティを使う

オーディオの長さ（尺）は、オーディオオブジェクトの「duration」プロパティを参照することで取得できます。「duration」プロパティは、長さを秒数で返します。

なお、オーディオデータが読み込まれていない状態で「duration」プロパティを読み出すと、「NaN」を返します。

スマートフォンでは一度、再生した後でないと、長さを取得できない場合があります。

SECTION-262

指定されたオーディオ形式が再生可能かどうか調べる

ここでは、指定されたオーディオ形式が再生可能かどうか調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <form>
      <input type="text" value="audio/wav" id="mtype"><br>
      <input type="button" value="再生可能か調べる">
    </form>
    <output></output>
  </body>
</html>
```

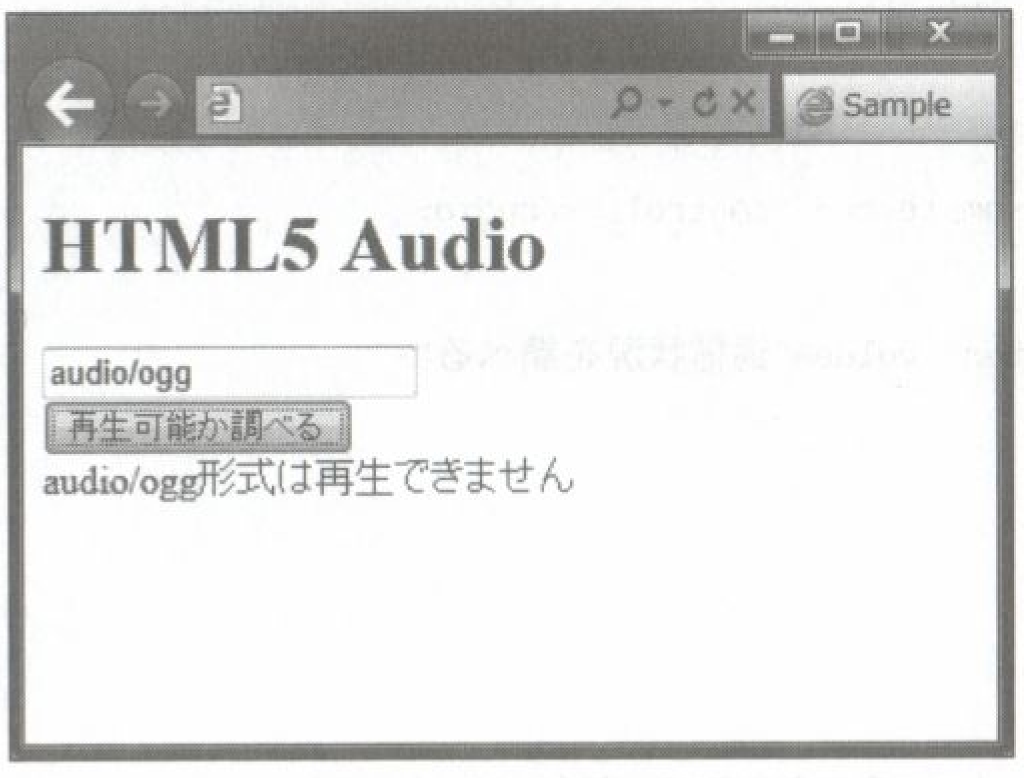
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オーディオオブジェクトを生成する
  var audioObj = new Audio();
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input")[1];
  btn.addEventListener("click", function(){
    // MIME Typeを読み出し
    var mt = document.getElementById("mtype").value;
    // 再生可能かどうか調べる
    var result = audioObj.canPlayType(mt);
    // 結果を表示
    ele.innerHTML = mt+"形式は再生できません";
    if (result === "maybe"){
      ele.innerHTML = mt+"形式は再生可能です";
    }
    if (result === "probably"){
      ele.innerHTML = mt+"形式は多分、再生できます";
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、再生できるかどうか調べたいオーディオファイルのMIMEタイプを入力後、ボタンをクリックすると、結果が表示されます。



再生できない場合は、次のように表示されます。



ONEPOINT **再生可能なオーディオファイルの形式を調べるには「canPlayType()」メソッドを使う**

HTML5 Audioでは、特定のファイル形式が再生できるかどうか調べることができます。再生可能かどうか調べるには、オーディオオブジェクトの「canPlayType()」メソッドを使います。「canPlayType()」メソッドのパラメータに、調べるファイルのMIMEタイプを指定します。

再生できる場合には「"maybe"」の文字を、再生できるかもしれない場合には「"probably"」の文字を返します。再生できない場合は、空文字が返されます。

なお、主なオーディオ形式のMIMEタイプは、次のようになります。

MIMEタイプ	説明
audio/mp3	MP3形式
audio/ogg	Ogg形式
audio/wav	WAVE形式
audio/aiff	AIFF形式
audio/mid	MIDI形式
audio/midi	MIDI形式

SECTION-263

オーディオファイルの読み込みに応じて変化する通信状態を調べる

ここでは、オーディオファイルの読み込みに応じて変化する通信状態を調べる方法について解説します。

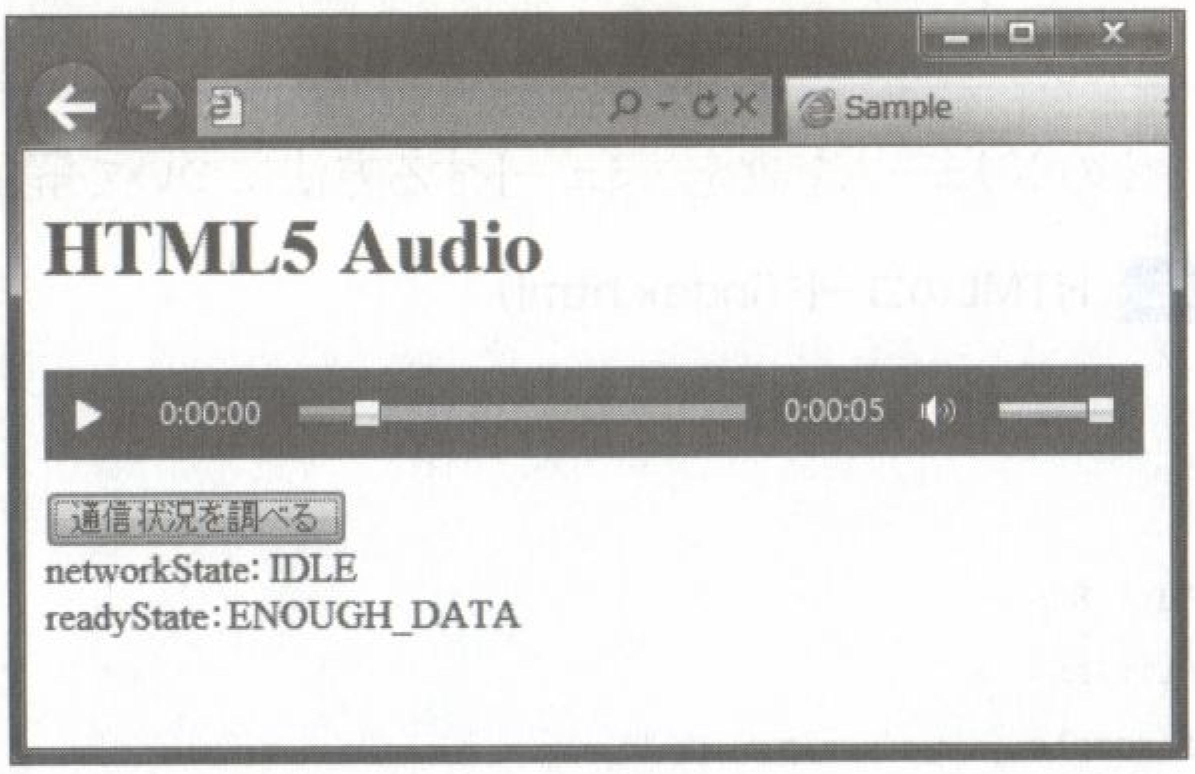
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="button" value="通信状況を調べる">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オーディオオブジェクトを生成する
  var audioObj = document.getElementsByTagName("audio")[0];
  // 通信状態を格納した配列を用意
  var nstat = ["EMPTY", "IDLE", "LOADING", "NO_SOURCE"];
  var rstat = ["NOTHING", "METADATA", "CURRENT_DATA", "FUTURE_DATA", "ENOUGH_DATA"];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input")[0];
  btn.addEventListener("click", function(){
    // ネットワーク状態を読み出し
    var s = audioObj.networkState;
    var rs = audioObj.readyState;
    // 結果を表示
    ele.innerHTML = "networkState:"+nstat[s];
    ele.innerHTML += "<br>readyState:"+rstat[rs];
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、現在の通信状態が画面に表示されます。



ONEPOINT 通信状態を調べるには「networkState」プロパティを使う

オーディオファイルの読み込みが行われると、通信状態が変化します。この通信状態がどのようになっているかは、オーディオオブジェクトの「networkState」プロパティを参照します。「networkState」プロパティには、次の表に示す値が入っています。

値	説明
0	ネットワークは空っぽ
1	アイドル中
2	読み込み中
3	ソース(ファイル/データ)がない

また、「networkState」プロパティ以外にも「readyState」プロパティでも状況を取得することができます。「readyState」プロパティには、次の表に示す値が入っています。

値	説明
0	データ受信前
1	メタデータの読み込み完了
2	現在の再生位置から再生できるデータをダウンロード
3	少し先まで再生できるデータをダウンロード
4	スムーズに再生できる充分なデータをダウンロード

COLUMN Windows Phoneの場合の「networkState」プロパティと「readyState」プロパティ

Windows Phoneではサンプルを実行すると、データの読み込み状態にかかわらず、「networkState」プロパティは「EMPTY」、「readyState」プロパティは「NOTHING」を返します。

SECTION-264

オーディオのボリュームを設定・ミュートにする

ここでは、オーディオのボリュームを設定・ミュートする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls loop></audio>
    <form>
      <input type="text" value="0.5" id="vol"><br>
      <input type="button" value="ボリュームを設定する"><br>
      <input type="button" value="ボリュームを読み出す"><br>
      <input type="button" value="ミュート(クリックで切り替え)">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オーディオオブジェクトを生成する
  var audioObj = document.getElementsByTagName("audio")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  // ボリューム設定を行うボタンにイベントを設定
  btn[1].addEventListener("click", function(){
    // 設定するボリューム値を読み出し
    var n = document.getElementById("vol").value;
    // オーディオのボリュームとして設定
    audioObj.volume = n;
    // 設定したことを示すメッセージを表示
    ele.innerHTML = "ボリュームを設定しました";
  }, false);
  // ボリュームを読み出すボタンにイベントを設定
  btn[2].addEventListener("click", function(){
    // 現在のボリューム値を読み出し
    var n = audioObj.volume;
```

```
      // 設定したことを示すメッセージを表示
      ele.innerHTML = "現在のボリューム:"+n;
    }, false);
    // ミュートを設定するボタンにイベントを設定
    btn[3].addEventListener("click", function(){
      // 現在のミュート状態を読み出し値を反転する
      var flag = audioObj.muted = !audioObj.muted;
      // 設定したことを示すメッセージを表示
      ele.innerHTML = "ミュート状態:"+flag;
    }, false);
  }, false);
```

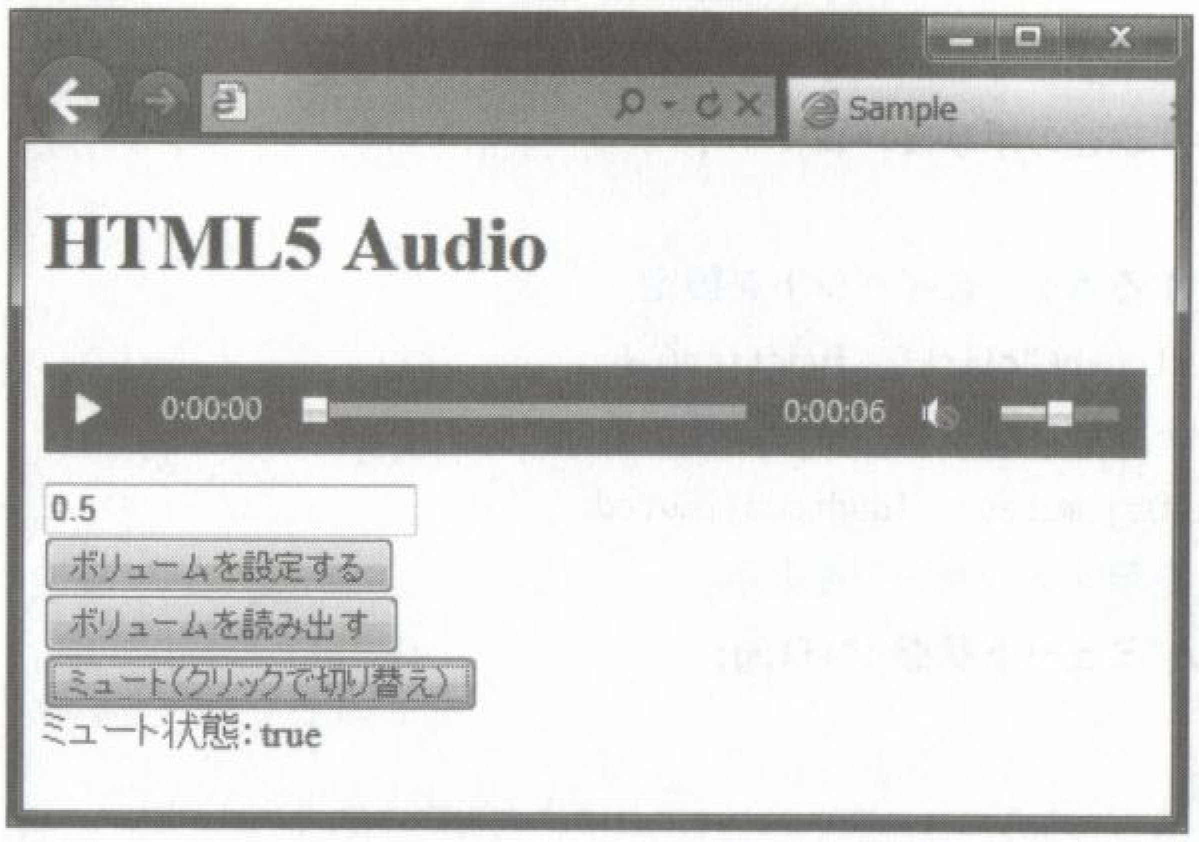
ブラウザでindex.htmlを表示し、テキストフィールドに設定したいボリューム値(「0.0」～「1.0」)を入力後、「ボリュームを設定する」ボタンをクリックすると、ボリュームが設定されます。



「ボリュームを読み出す」ボタンをクリックすると、現在のボリューム値が表示されます。



「ミュート(クリックで切り替え)」ボタンは、クリックするたびに、ミュートのON/OFFが切り替わります。



ONEPOINT

ボリュームを設定するには「volume」プロパティ、ミュートするには「muted」プロパティを使う

HTML5 Audioでボリューム(音量)を設定するには、オーディオオブジェクトの「volume」プロパティに値を代入します。設定できる値は、「0.0」～「1.0」の範囲になります。この範囲を超えた場合、自動的に値を調整して設定される場合もあります(負数の場合は設定されない場合がある)。また、現在のボリュームを読み出すには、オーディオオブジェクトの「volume」プロパティを参照します。

ボリュームの調整ではなく、一時的に音を消す(ミュート/消音)場合には、オーディオオブジェクトの「muted」プロパティに「true」を設定します。再度、音を出すようにするには「false」を設定します。

なお、ボリューム関係は、スマートフォンでは読み出すことはできても、設定することができない場合があります。その際、設定してもエラーは発生しません。

関連項目 ▶▶▶

- オーディオのボリュームが変更された際のイベントを取得する ……………………………… p.635
- Web Audioでボリュームを指定する……………………………………………………………… p.855

SECTION-265

オーディオのコントローラーの表示/非表示を設定する

ここでは、オーディオのコントローラーの表示/非表示を設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="button" value="表示する">
      <input type="button" value="非表示にする">
    </form>
  </body>
</html>
```

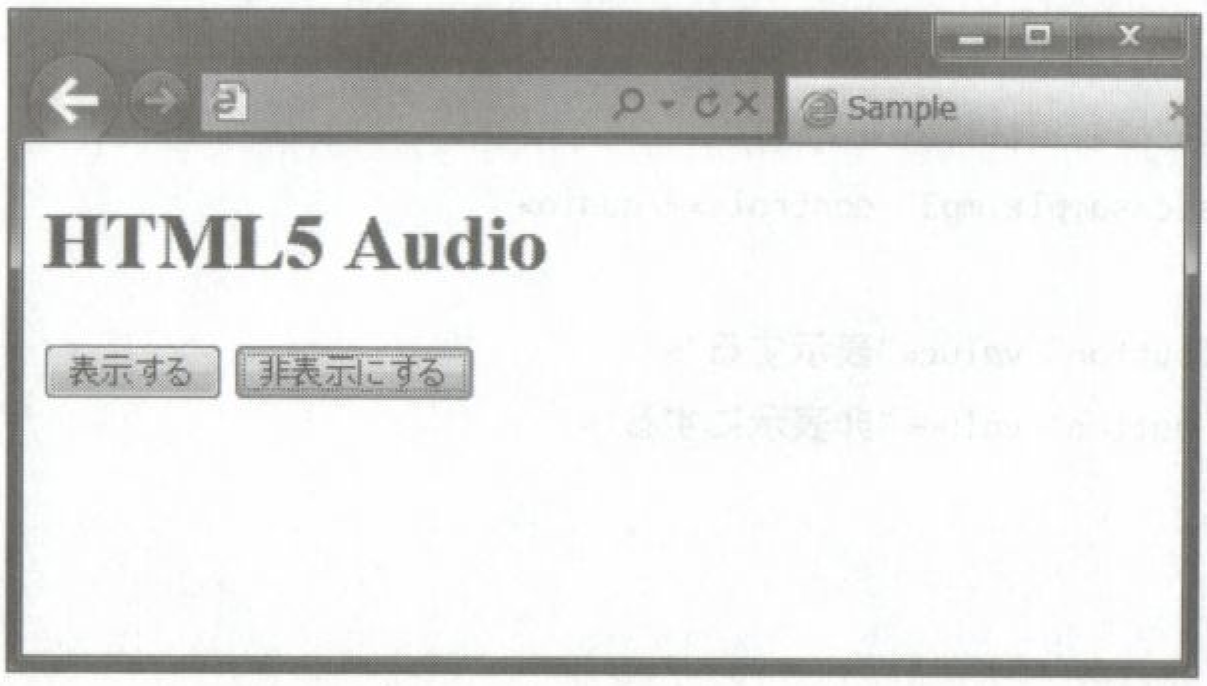
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // オーディオオブジェクトを読み出し
  var audioObj = document.getElementsByTagName("audio")[0];
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // コントローラーを表示する
    audioObj.controls = true;
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // コントローラーを非表示にする
    audioObj.controls = false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、最初の状態ではコントローラーは表示されています。



「非表示にする」ボタンをクリックすると、コントローラーが消えます。なお、ブラウザによっては、「auido」要素の枠が残る場合があります。



ONEPOINT オーディオのコントローラーの表示制御を行うには「controls」プロパティを使う

「audio」要素では、ユーザーにオーディオ操作させる際にコントローラーを表示するための「controls」属性が用意されています。スマートフォンやタブレットによっては、「controls」属性を指定しないと再生することができない(表示されない)場合があります。また、コントローラーは表示する必要があるが、再生中は消しておきたい場合もあります。

コントローラーの表示制御は、オーディオオブジェクトの「controls」プロパティに「true」、または、「false」を代入することで行います。「true」にするとコントローラーが表示され、「false」にするとコントローラーは非表示になります。

SECTION-266

オーディオデータを読み込む

ここでは、オーディオデータを読み込む方法について解説します。

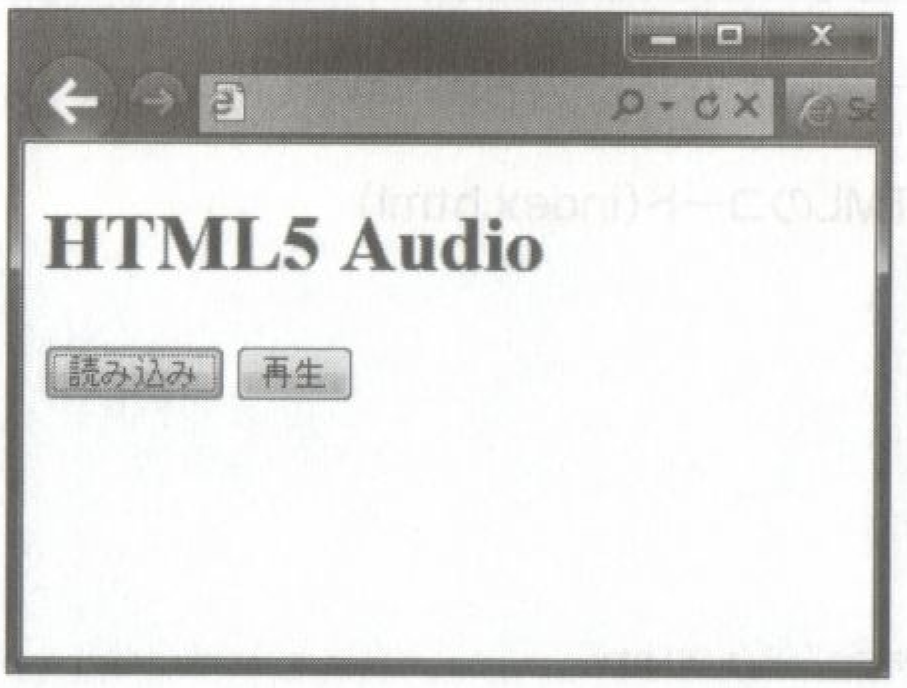
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <form>
      <input type="button" value="読み込み">
      <input type="button" value="再生">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // オーディオオブジェクトを生成
  var audioObj = new Audio();
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // オーディオファイルを読み込み
    audioObj.src = "music/sample.mp3";
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // オーディオを再生する
    audioObj.play();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「読み込み」ボタンをクリックすると、オーディオファイルが読み込まれます。オーディオファイルが読み込まれていない時点では「再生」ボタンをクリックしても再生されません。「読み込み」ボタンをクリックした後であれば、オーディオは再生されます。



ONEPOINT　オーディオデータを読み込ませるには「src」プロパティを使う

ゲームなどでは、状況に合わせてオーディオファイルを再生したい場合があります。このような場合、オーディオオブジェクトの「src」プロパティに再生したいオーディオファイルのURLを指定することで、動的に再生されるオーディオを変えることができます。

関連項目 ▶▶▶

● Web Audioでファイルを読み込んで演奏する ………… p.852

SECTION-267

オーディオの再生状態を調べる

ここでは、オーディオの再生状態を調べる方法について解説します。

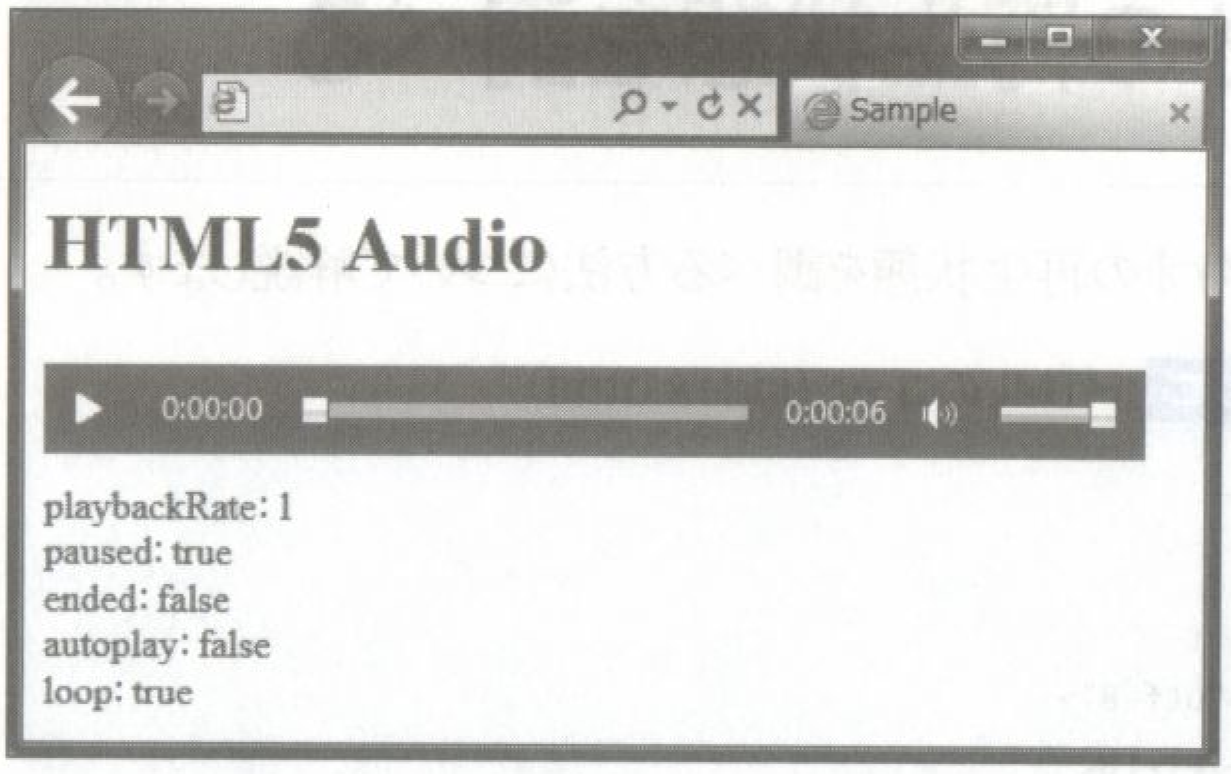
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls loop></audio>
    <br>
    <output></output>
  </body>
</html>
```

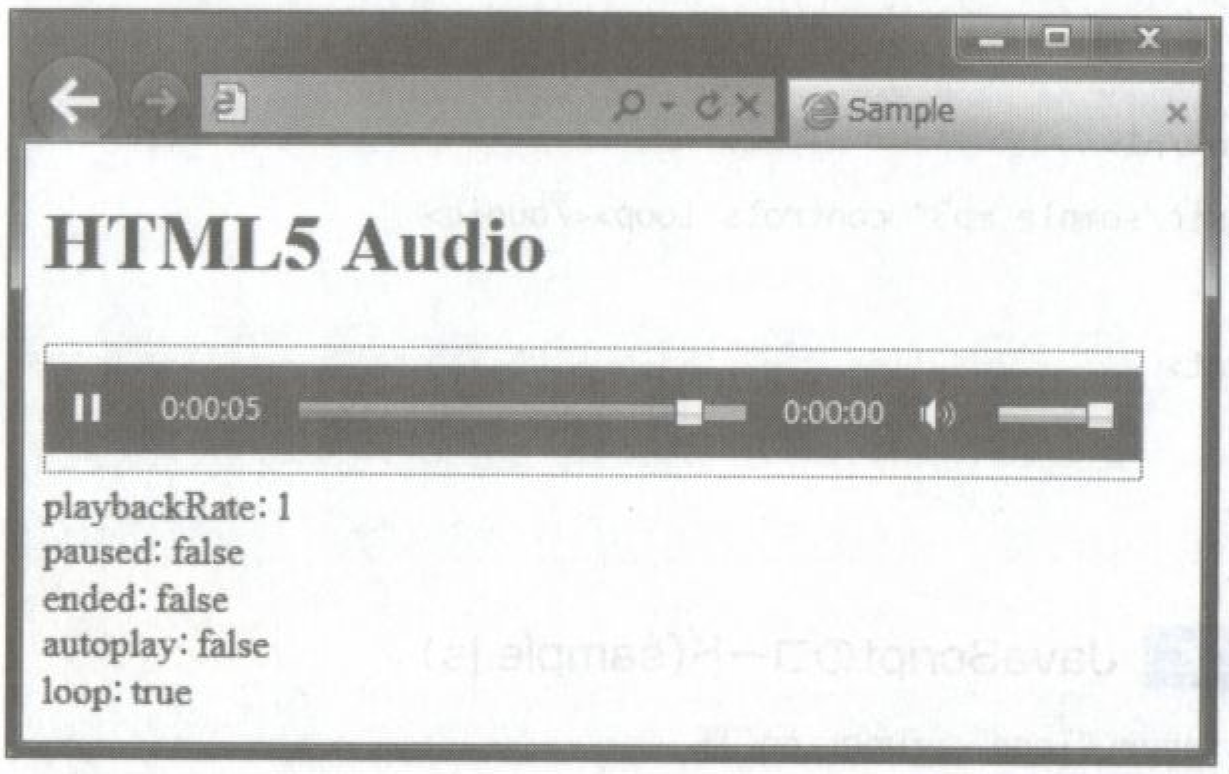
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オーディオ要素を読み出す
  var audioObj = document.getElementsByTagName("audio")[0];
  // タイマーを使ってオーディオの状態を監視する
  setTimeout(function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生速度を表示
    ele.innerHTML = "playbackRate:"+audioObj.playbackRate;
    // 一時停止状態を表示
    ele.innerHTML += "<br>paused:"+audioObj.paused;
    // 再生終了状態を表示
    ele.innerHTML += "<br>ended:"+audioObj.ended;
    // 自動再生状態を表示
    ele.innerHTML += "<br>autoplay:"+audioObj.autoplay;
    // ループ(繰り返し再生)状態を表示
    ele.innerHTML += "<br>loop:"+audioObj.loop;
    // タイマー処理を継続
    setTimeout(arguments.callee, 250);
  }, 250);
}, false);
```

ブラウザでindex.htmlを表示すると、オーディオの状態がコントローラーの下に表示されます。



オーディオを再生すると、プロパティの状態が変化します。



ONEPOINT

オーディオの状態を取得するには「currentTime」プロパティや「paused」プロパティなどを使う

HTML5 Audioでは、さまざまなプロパティがあります。その中で再生状態を示すプロパティは次の表のようになります。プロパティは、どのブラウザでも読み出すことができます。

プロパティ	説明
currentTime	現在の再生時間(再生ヘッドの位置)
duration	オーディオの長さ
paused	停止しているかどうか。停止していれば「true」、再生時は「false」
playbackRate	再生速度(等倍速だと「1.0」、2倍速なら「2.0」)
ended	再生が終了しているか。終了したら「true」、そうでなければ「false」
autoplay	自動再生するかどうか。自動再生する場合は「true」、そうでない場合は「false」
loop	繰り返し(ループ)再生するかどうか。「true」なら繰り返し、「false」なら繰り返さない。ただし、Firefoxではloopが機能しない

SECTION-268

オーディオの再生速度を変更する

ここでは、オーディオの再生速度を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <form>
      <input type="button" value="1倍速">
      <input type="button" value="2倍速">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // オーディオオブジェクトを生成
  var audioObj = document.getElementsByTagName("audio")[0];
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // 1倍速で再生する
    audioObj.playbackRate = 1.0;
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // 2倍速で再生する
    audioObj.playbackRate = 2.0;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、再生速度が変更されます。



ONEPOINT **再生速度を変更するには「playbackRate」プロパティを使う**

IE9以降やGoogle Chrome、Safariでは、オーディオの再生速度を変更することができます。この場合、オーディオオブジェクトの「playbackRate」プロパティに再生速度の倍率を指定します。「1.0」であれば等倍速に、「0.5」なら半分の速度(スロー再生)、「2.0」なら倍速再生になります。あまり大きい数値や小さい数値を指定すると、指定した通りの速度で再生されないことがあります。IE9の場合、「playbackRate」プロパティに値を設定してから実際に反映されるまでに数秒かかることもあります。また、IE9ではオーディオがループ再生されている場合、オーディオの最初に戻ると、「playbackRate」プロパティに設定した値は反映されずデフォルト値に戻ります。

デフォルトの再生速度を設定する場合には、「defaultPlaybackRate」プロパティに再生速度の倍率を指定します。「defaultPlaybackRate」プロパティを設定しておけば、IE9でもループ再生で最初に戻ったときに設定した速度で映像が再生されます。

なお、未対応のブラウザやスマートフォン(iOS/Android/Windows Phone)に設定した場合はエラーにはならず、再生速度も変化しません。

IE 9 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-269

オーディオの再生時のイベントを取得する

ここでは、オーディオの再生時のイベントを取得する方法について解説します。

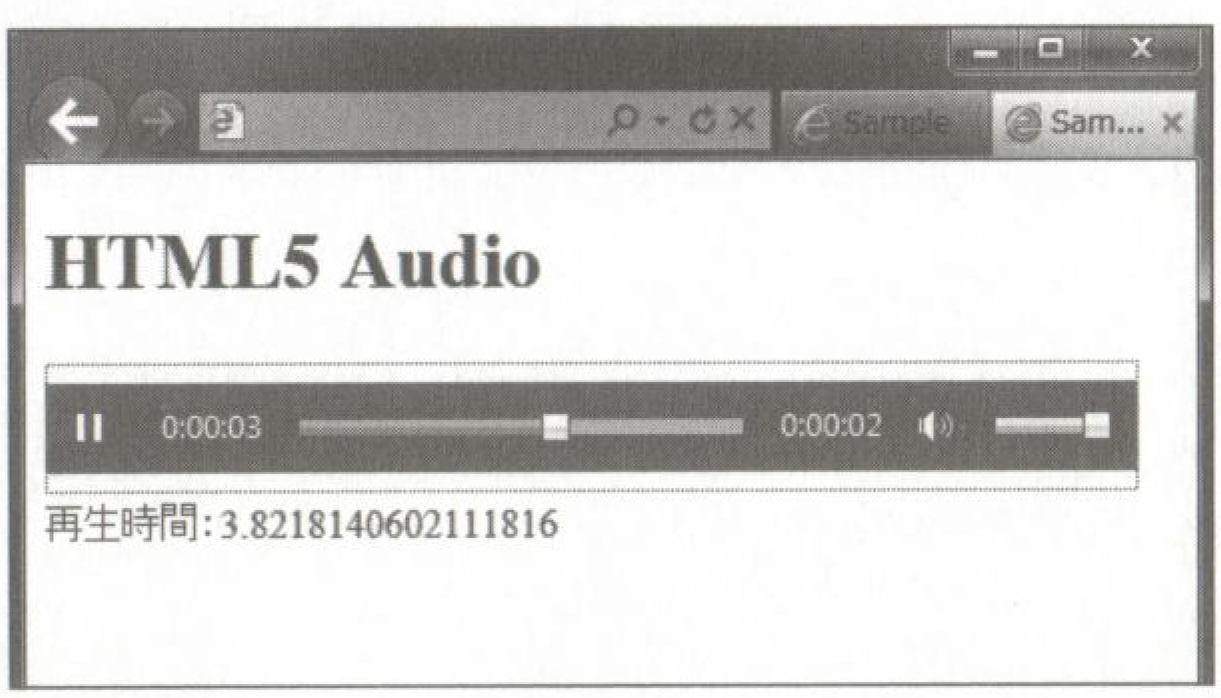
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
    <br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オーディオ要素を読み出す
  var audioObj = document.getElementsByTagName("audio")[0];
  // オーディオにイベントリスナーを設定
  audioObj.addEventListener("timeupdate", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生時間を表示
    ele.innerHTML = "再生時間:"+this.currentTime;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、オーディオを再生すると、コントローラーの下に再生時間が表示されます。また、再生していない状態でも、コントローラー上のスライダーをドラッグすると、再生時間が表示されます。



ONEPOINT **オーディオが再生された際に処理を設定するには「timeupdate」イベントを使う**

HTML5 Audioでは、現在の再生時間を示す「currentTime」プロパティの値が変化した場合に、「timeupdate」イベントが発生します。「play()」メソッドを使ったり、コントローラーで再生したりした際に発生するのはもちろん、コントローラーで再生時間を示すスライダーをドラッグした場合にも「timeupdate」イベントは発生します。

「timeupdate」イベントの発生間隔は、ブラウザや動作環境(OSやバージョン)によって異なります。Firefoxのように頻繁に発生するブラウザもあれば、Safariのように1秒間に数回しか発生しないブラウザもあります。

より短い間隔で再生時間を取得したい場合には、「setTimeout()」メソッドか「setInterval()」メソッドなどのタイマー処理行うメソッドを使う必要があります。

関連項目 ▶▶▶

- イベントハンドラを設定する ………… p.396
- イベントリスナーを設定する ………… p.398
- オーディオを再生・停止する ………… p.610
- オーディオの再生が終了した際のイベントを取得する ………… p.633
- オーディオのボリュームが変更された際のイベントを取得する ………… p.635

SECTION-270

オーディオの再生が終了した際のイベントを取得する

ここでは、オーディオの再生が終了した際のイベントを取得する方法について解説します。

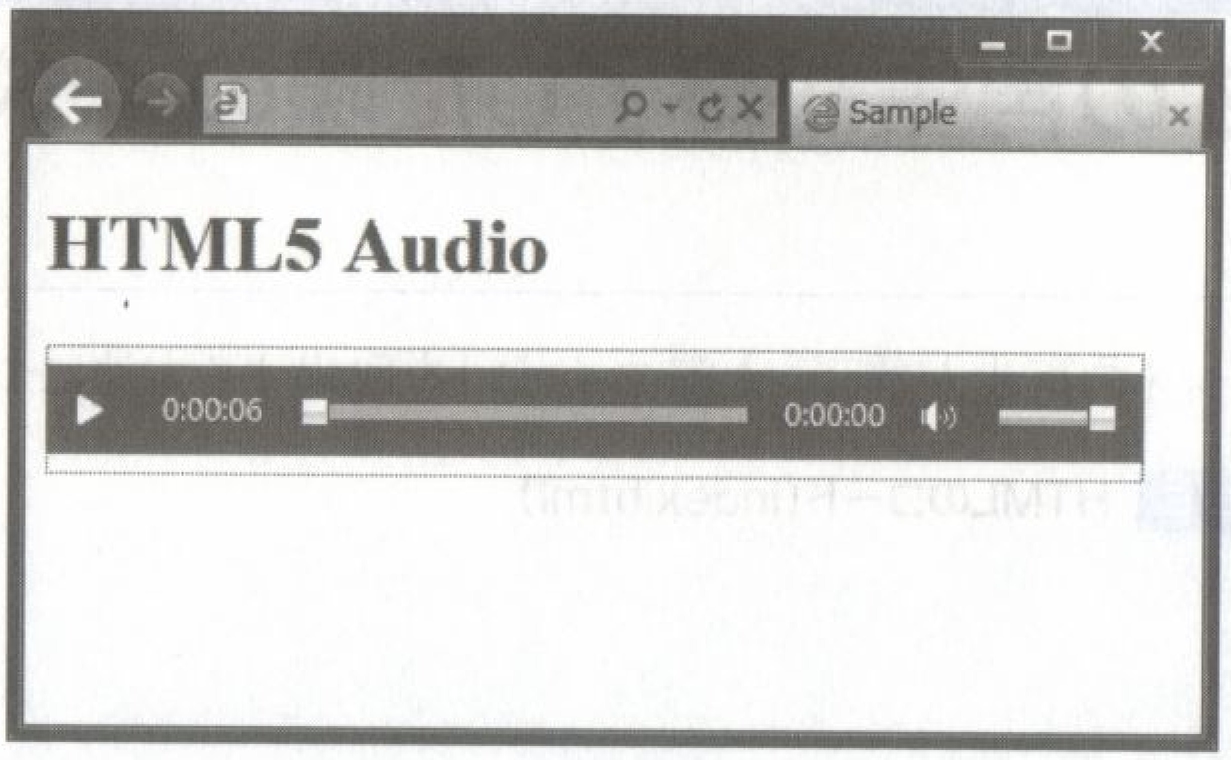
SAMPLE CODE HTMLのコード(index.html)

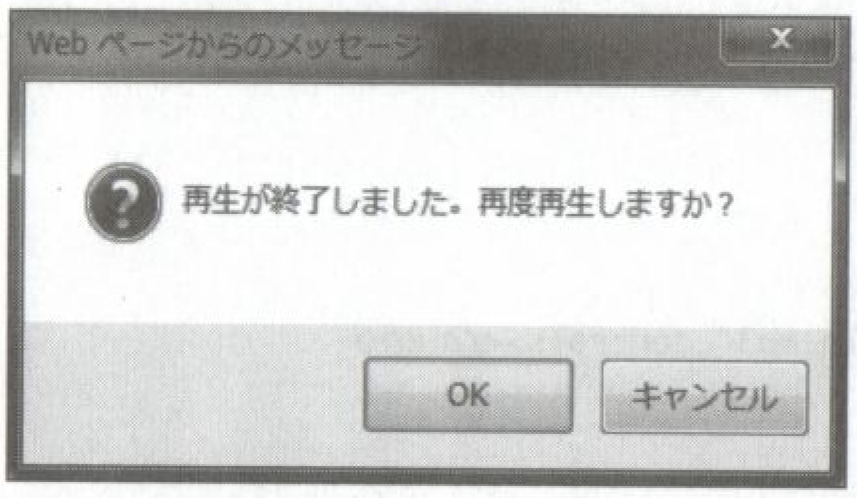
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls></audio>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // オーディオ要素を読み出す
  var audioObj = document.getElementsByTagName("audio")[0];
  // オーディオにイベントリスナーを設定
  audioObj.addEventListener("ended", function(){
    var flag = confirm("再生が終了しました。再度再生しますか？");
    if (flag === true){
      // 再生時間を0(最初)に設定する
      this.currentTime = 0;
      // 再生する
      this.play();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、オーディオの再生が終了すると、再生を再度、行うかどうかのダイアログが表示されます。「OK」ボタンをクリックすると再度オーディオが再生され、「キャンセル」のボタンをクリックするとそこで再生は終了して元の画面に戻ります。





ONEPOINT

オーディオの再生が終了した際の処理を設定するには「ended」イベントを使う

HTML5 Audioでは、再生が終了すると、「ended」イベントが発生します。ゲームのようにオープニングでオーディオの再生が終了したら他のページに移動したり、次の処理を行う場合に利用できます。ただし、オーディオで繰り返し再生を行う「loop」属性が設定されていると、「ended」イベントは発生しません。なお、Firefoxでは「loop」属性を設定しても無効なので、常に「ended」イベントが発生することになります。Firefoxで繰り返し再生する場合は、「ended」イベントを検知したら、「currentTime」プロパティに「0」を設定し、再度、「play()」メソッドを使って最初から再生する必要があります。

関連項目 ▶▶▶

- イベントハンドラを設定する ………… p.396
- イベントリスナーを設定する ………… p.398
- オーディオを再生・停止する ………… p.610
- オーディオの再生時のイベントを取得する ………… p.631
- オーディオのボリュームが変更された際のイベントを取得する ………… p.635

SECTION-271

オーディオのボリュームが変更された際のイベントを取得する

ここでは、オーディオのボリュームが変更された際のイベントを取得する方法について解説します。

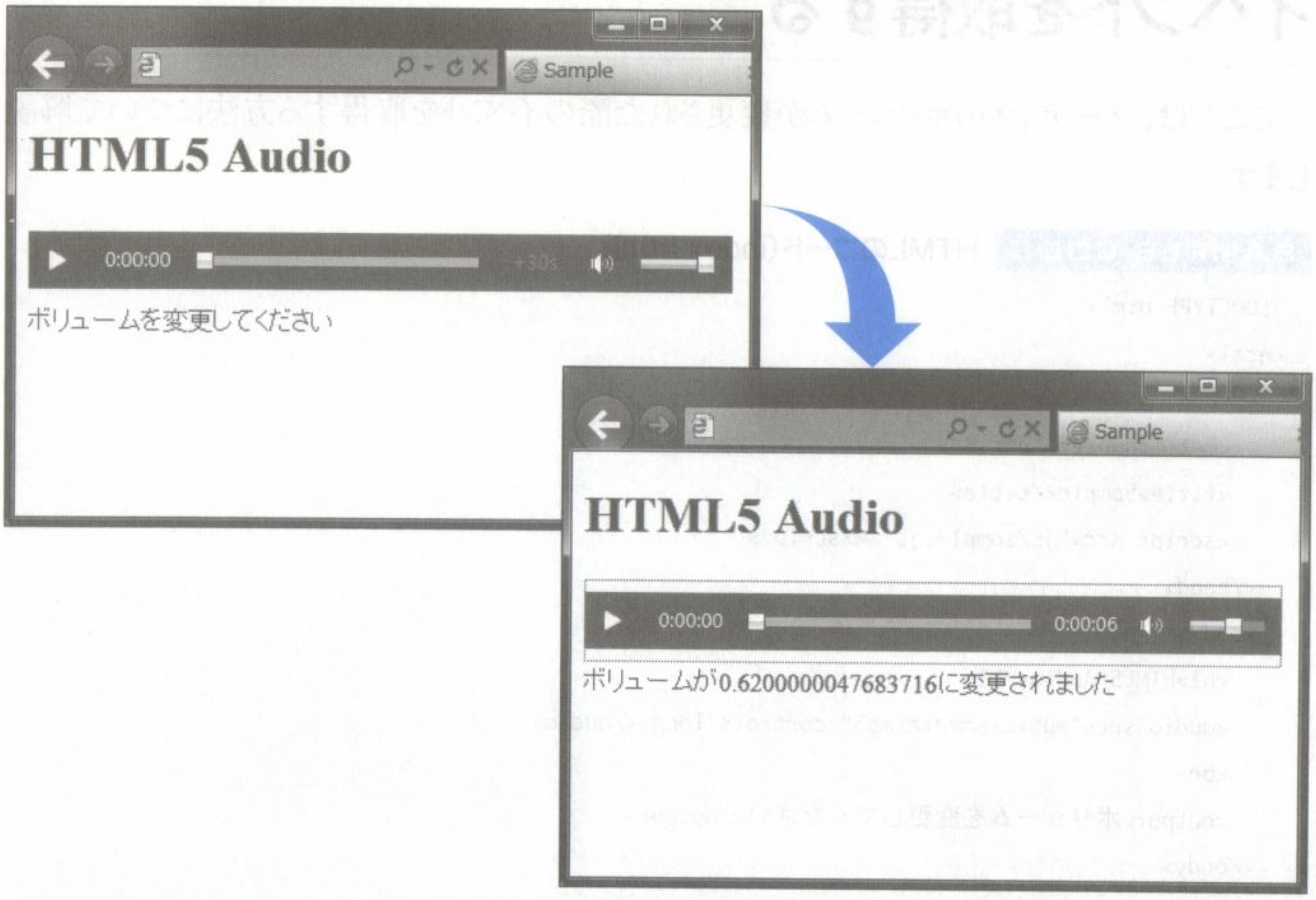
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Audio</h1>
    <audio src="music/sample.mp3" controls loop></audio>
    <br>
    <output>ボリュームを変更してください</output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オーディオオブジェクトを生成する
  var audioObj = document.getElementsByTagName("audio")[0];
  // オーディオにイベントを割り当てる
  audioObj.addEventListener("volumechange", function(){
    // ボリュームを読み出し
    var n = audioObj.volume;
    // ボリュームが変更された事を示すメッセージを表示
    ele.innerHTML = "ボリュームが"+n+"に変更されました";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、コントローラーのボリュームスライダーをドラッグすると、リアルタイムに変更されたボリュームの値が表示されます。



ONEPOINT　ボリューム変更時の処理を行うには「volumechange」イベントを使う

HTML5 Audioでボリューム（音量）が変更されると、「volumechange」イベントが発生します。「volumechange」イベントが発生した後のボリュームの値は、変更後のものになります。

なお、スマートフォン（iPhone/iPad/Android/Windows Phone）では、本体のボリュームを変更しても、「volumechange」イベントは発生しません。

関連項目 ▶▶▶

- イベントハンドラを設定する …… p.396
- イベントリスナーを設定する …… p.398
- オーディオのボリュームを設定・ミュートにする …… p.620
- オーディオの再生時のイベントを取得する …… p.631
- オーディオの再生が終了した際のイベントを取得する …… p.633

「video」要素にアクセスする

ここでは、「video」要素にアクセスする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="再生">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input")[0];
  btn.addEventListener("click", function(){
    // 最初のビデオ要素をピックアップ
    var videoObj = document.getElementsByTagName("video")[0];
    // ビデオを再生する
    videoObj.play();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、映像が再生されます。また、コントローラーでも映像の再生・停止を行うことができます。なお、スマートフォン(iPhone/Android 2/Windows Phone)では再生はフルスクリーンで行われます。iPad/Android 3/Android 4では画面内で、そのまま再生されます。



ONEPOINT 映像にアクセスするには「video」要素を使う

HTML5では「video」要素を使うことで、プラグインを使わなくても手軽にページ内に映像を埋め込むことができます。ただし、映像の形式やコーデックは多くあり、状況によってサポートされる形式が異なる場合があります。

最もシンプルな書き方は、サンプルのように、「video」要素の「src」属性に再生する映像ファイルのURLを指定することです。しかし、この記述方法では映像形式に対応していないブラウザや、映像再生自体できないデバイスに対応できません。このため、次のように記述することで複数の映像形式に対応することができます。

```
<video>
  <source src="映像ファイル、その1">
  <source src="映像ファイル、その2">
  <source src="映像ファイル、その3">
  <p>HTML5 Videoに対応していない場合のメッセージなど</p>
</video>
```

指定した映像にJavaScriptからアクセスするには、「document.getElementById()」メソッドや「document.getElementsByTagName()」メソッド、「document.querySelector()」メソッド、「document.querySelectorAll()」メソッドなどを使います。

映像データの読み込みには時間がかかることが多いため、「poster」属性を使って映像の代わりに画像を表示しておくこともできます。

COLUMN FirefoxやOperaでの映像の再生について

FirefoxやOperaではOgg/Ogv形式でないと、映像が再生されません。本書のサンプルでは、MP4形式の映像ファイルを使っているため、そのままでは映像が再生されず、動作の確認を行うことができません。FirefoxやOperaで動作確認を行う際は、別途、Ogg/Ogv形式の映像を用意し、サンプルのコードを適宜、書き換えてください。

SECTION-273

映像を再生・停止する

ここでは、映像を再生・停止する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="再生">
      <input type="button" value="停止">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ビデオ要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  btn[0].addEventListener("click", function(){
    // 映像を再生
    videoObj.play();
  }, false);
  btn[1].addEventListener("click", function(){
    // 映像の再生を停止
    videoObj.pause();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、再生ボタンをクリックすると、映像が再生されます。停止ボタンをクリックすると、再生が停止します。



ONEPOINT 再生を行うには「play()」メソッド、停止を行うには「pause()」メソッドを使う

HTML5 Videoを再生するには、ビデオオブジェクトの「play()」メソッドを使います。「play()」メソッド実行時に映像ファイルが読み込まれていない場合でもエラーにはならず、再生可能になった段階で再生が行われます。

映像の再生を停止するには、「pause()」メソッドを使います。「pause()」メソッドは再生を停止するだけで、巻き戻し処理などは行いません。このため、「pause()」メソッド実行後に再度、「play()」メソッドを実行すると停止したところから再生が開始されます。

スマートフォン(iPhone/Android 2/Windows Phone)では「play()」メソッドを実行すると、フルスクリーンで再生が行われます。iPad/Android 3/Android 4では画面内で、そのまま再生されます。

関連項目 ▶▶▶

- 映像の再生時のイベントを取得する ……… p.661
- 映像の再生が終了した際のイベントを取得する ……… p.663
- 映像をフルスクリーン表示にする ……… p.796
- カメラからの映像を表示する ……… p.667

映像の再生時間を取得・設定する

ここでは、映像の再生時間を取得・設定する方法について解説します。

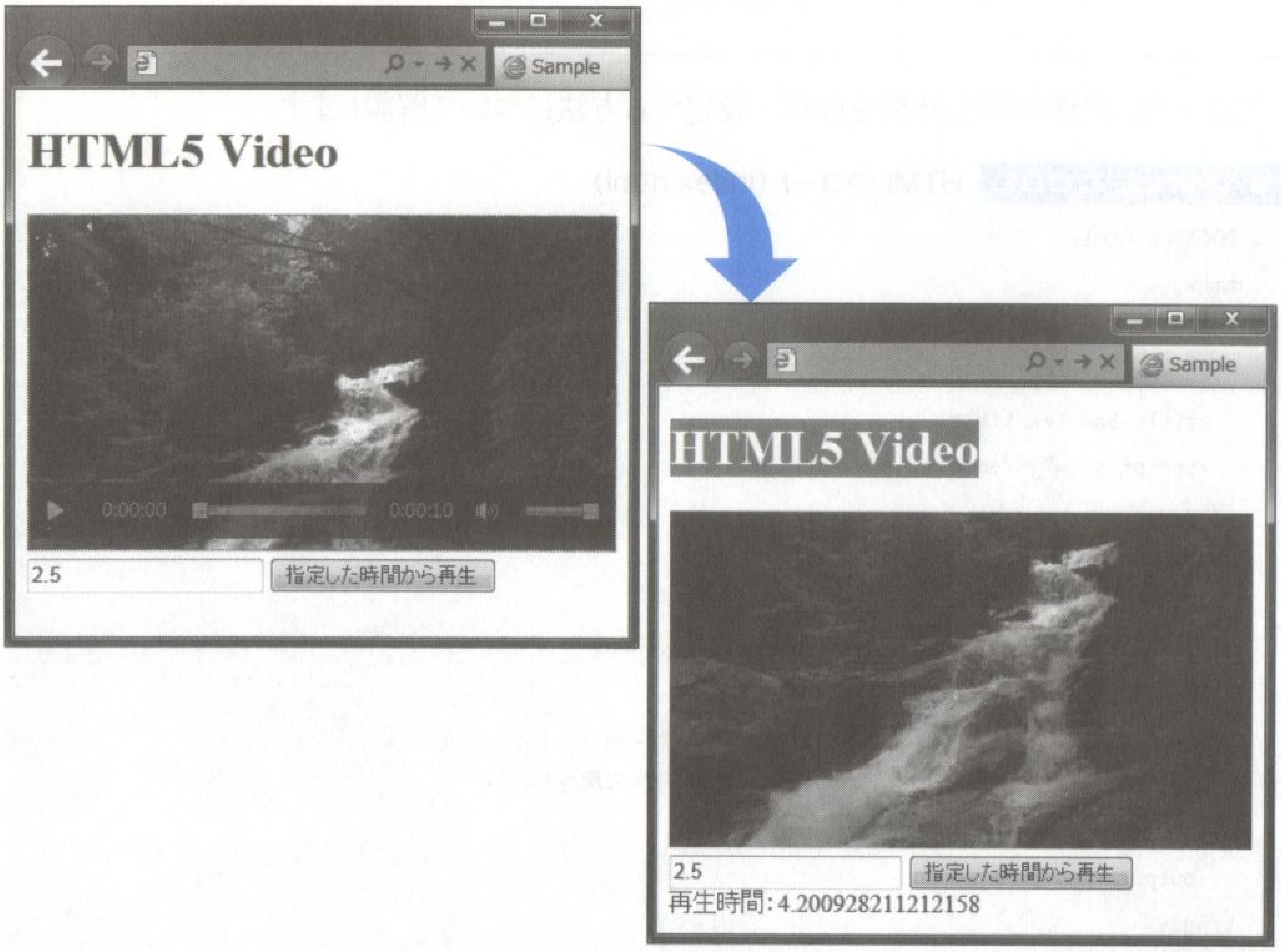
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="text" value="2.5" id="startTime">
      <input type="button" value="指定した時間から再生">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ビデオ要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  btn[1].addEventListener("click", function(){
    // 設定時間を読み出し
    var time = document.getElementById("startTime").value;
    // 時間を設定
    videoObj.currentTime = time;
    // 映像を再生
    videoObj.play();
  }, false);
  // ビデオ要素にイベントを割り当て
  videoObj.addEventListener("timeupdate", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生時間を表示
    ele.innerHTML = "再生時間：" + this.currentTime;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、開始時間を入力した後で、再生ボタンをクリックすると、映像が指定した時間から再生され、画面に再生時間が表示されます。



ONEPOINT

再生開始時間の設定と再生時間の取得には「currentTime」プロパティを使う

現在の再生時間を取得するには、ビデオオブジェクトの「currentTime」プロパティを参照します。「currentTime」プロパティの内容は再生されている時間であり、「5.963」のようにミリ秒を含む値になります。

指定した時間から再生する場合には、「currentTime」プロパティに再生を開始したい時間を設定します。なお、再生中でも、停止中でも、「currentTime」プロパティに値を設定することができます。

COLUMN

スマートフォンの場合の「currentTime」プロパティ

Android 2では、「currentTime」プロパティに秒数を設定した場合、コントローラー上には反映されることがありますが、映像は最初から再生されます。

また、iOSでは一度、映像を再生した後でないと、「currentTime」プロパティに秒数を設定しても反映されません。

Windows Phone（7.5）では、「currentTime」プロパティに秒数を設定しても反映されず、最初から再生が行われます。

SECTION-275

映像の長さを調べる

ここでは、映像の長さを調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="長さを調べる">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input")[0];
  // ビデオオブジェクトを生成
  var videoObj = document.getElementsByTagName("video")[0];
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(){
    // オーディオファイルの長さ(尺)を変数に代入
    var sec = videoObj.duration;
    document.getElementsByTagName("output")[0].innerHTML = sec+"秒";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、映像の長さが表示されます。



ONEPOINT 映像の長さを取得するには「duration」プロパティを使う

映像の長さ(尺)は、ビデオオブジェクトの「duration」プロパティを参照することで取得できます。「duration」プロパティは、長さを秒数で返します。ただし、映像が読み込まれていない場合は、「NaN」になります。

また、同じ映像でも、ブラウザによって違う長さを返してくる場合があります。このため、1/10秒より精度を必要とする場合は注意が必要です。映像のフォーマットを変換した場合にも、同様に、映像の長さが変わってしまう場合があります。

スマートフォン(iOS/Android)では、映像が再生されていない場合(映像データが読み込まれていない場合)、「duration」プロパティは「NaN」を返します。Windows Phoneでは、映像データが読み込まれていない場合は「NaN」を返しますが、読み込まれた後でも映像の長さが返されません。このため、映像の長さを取得することはできません。

SECTION-276

指定した形式の映像が再生可能かどうか調べる

ここでは、指定された映像の形式が再生可能かどうか調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <form>
      <input type="text" value="video/mpeg" id="mtype"><br>
      <input type="button" value="再生可能か調べる">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // video要素を生成する
  var videoObj = document.createElement("video");
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input")[1];
  btn.addEventListener("click", function(){
    // MIME Typeを読み出し
    var mt = document.getElementById("mtype").value;
    // 再生可能かどうか調べる
    var result = videoObj.canPlayType(mt);
    // 結果を表示
    ele.innerHTML = mt+"形式は再生できません";
    if (result === "maybe"){
      ele.innerHTML = mt+"形式は再生可能です";
    }
    if (result === "probably"){
      ele.innerHTML = mt+"形式は多分、再生できます";
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、再生できるかどうか調べたい映像ファイルのMIMEタイプを入力後、ボタンをクリックすると、結果が表示されます。



再生できない場合は、次のように表示されます。



ONEPOINT　映像ファイルの形式が再生可能か調べるには「canPlayType()」メソッドを使う

HTML5 Videoでは、特定のファイル形式が再生できるかどうかを調べることができます。再生可能かどうか調べるにはビデオオブジェクトの「canPlayType()」メソッドを使います。「canPlayType()」メソッドのパラメータに、調べるファイルのMIMEタイプを指定します。

再生できる場合には「"maybe"」の文字を、再生できるかもしれない場合には「"probably"」の文字を返します。再生できない場合は、空文字が返されます。

なお、主な映像形式のMIMEタイプは、次の表のようになります。

MIMEタイプ	説明
video/mpeg	MPEG形式
video/mp4	MPEG 4形式
video/quicktime	QuickTime形式
video/ogg	Ogg(Ogv)形式
video/3gpp	3GPP形式
video/webm	WebM形式

SECTION-277

映像ファイルの読み込みに応じて変化する通信状態を調べる

ここでは、映像ファイルの読み込みに応じて変化する通信状態を調べる方法について解説します。

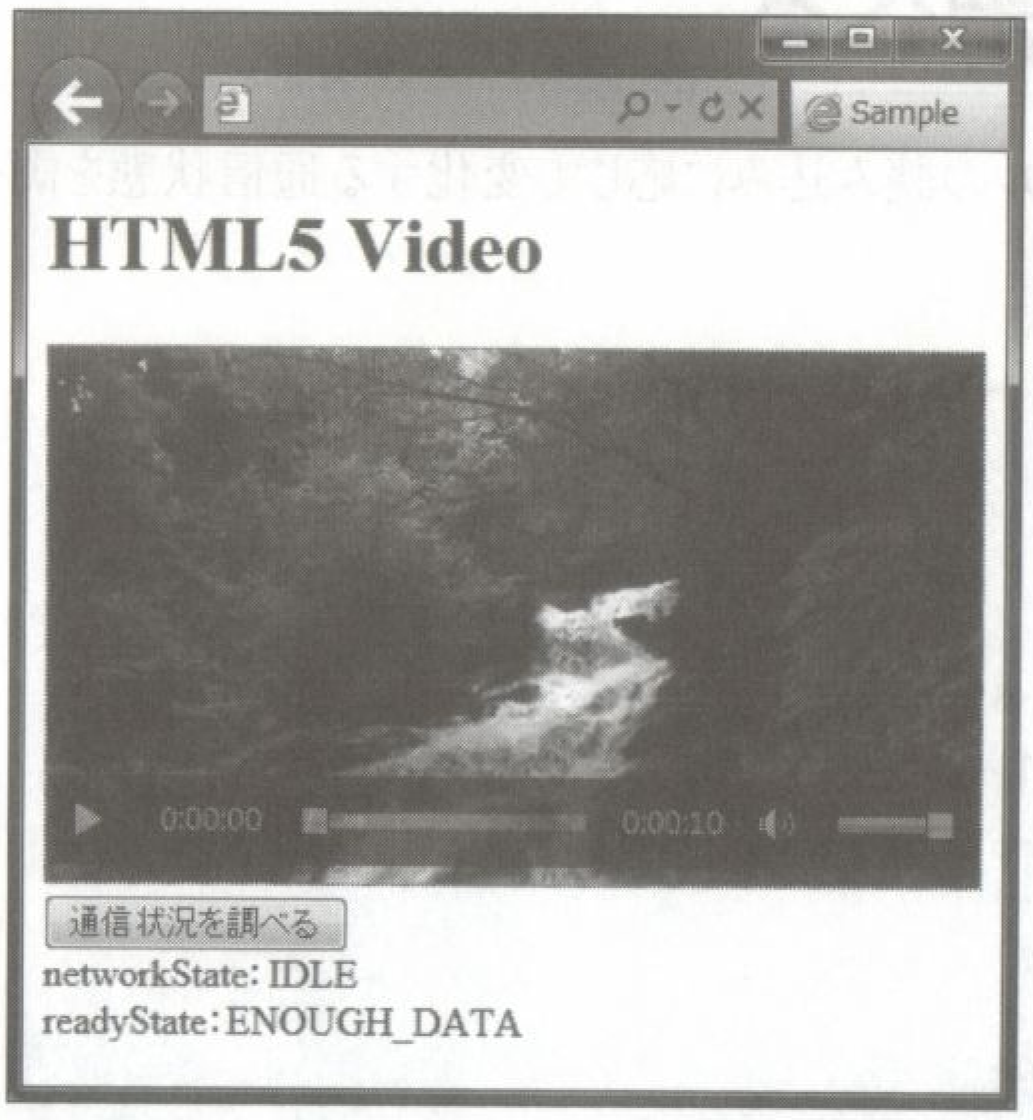
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="通信状況を調べる">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ビデオオブジェクトを生成する
  var videoObj = document.getElementsByTagName("video")[0];
  // 通信状態を格納した配列を用意
  var nstat = ["EMPTY", "IDLE", "LOADING", "NO_SOURCE"];
  var rstat = ["NOTHING", "METADATA", "CURRENT_DATA", "FUTURE_DATA", "ENOUGH_DATA"];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input")[0];
  btn.addEventListener("click", function(){
    // ネットワーク状態を読み出し
    var s = videoObj.networkState;
    var rs = videoObj.readyState;
    // 結果を表示
    ele.innerHTML = "networkState:"+nstat[s];
    ele.innerHTML += "<br>readyState:"+rstat[rs];
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、現在の通信状態が画面に表示されます。



ONEPOINT 通信状態を調べるには「networkState」プロパティを使う

映像ファイルの読み込みが行われると、通信状態が変化します。この通信状態がどのようになっているかは、ビデオオブジェクトの「networkState」プロパティを参照します。「networkState」プロパティには、次の表に示す値が入っています。

値	説明
0	ネットワークは空っぽ
1	アイドル中
2	読み込み中
3	ソース(ファイル/データ)がない

また、「networkState」プロパティ以外にも、「readyState」プロパティでも状況を取得することができます。「readyState」プロパティには、次の表に示す値が入っています。

値	説明
0	データ受信前
1	メタデータの読み込み完了
2	現在の再生位置から再生できるデータをダウンロード
3	少し先まで再生できるデータをダウンロード
4	スムーズに再生できる十分なデータをダウンロード

SECTION-278

映像のボリュームを設定・ミュートする

ここでは、映像のボリュームを設定・ミュートする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

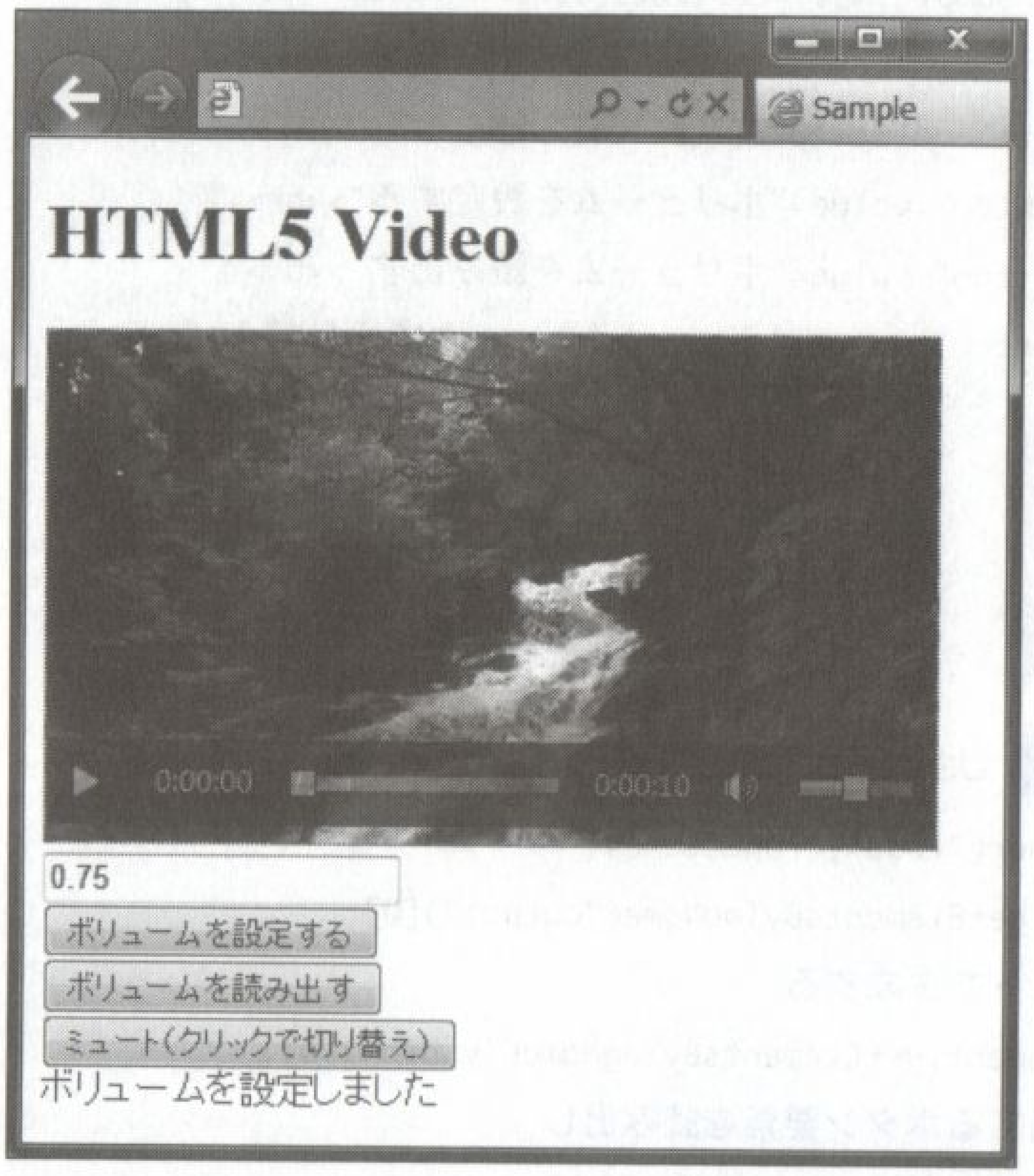
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls loop></video>
    <form>
      <input type="text" value="0.5" id="vol"><br>
      <input type="button" value="ボリュームを設定する"><br>
      <input type="button" value="ボリュームを読み出す"><br>
      <input type="button" value="ミュート(クリックで切り替え)">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

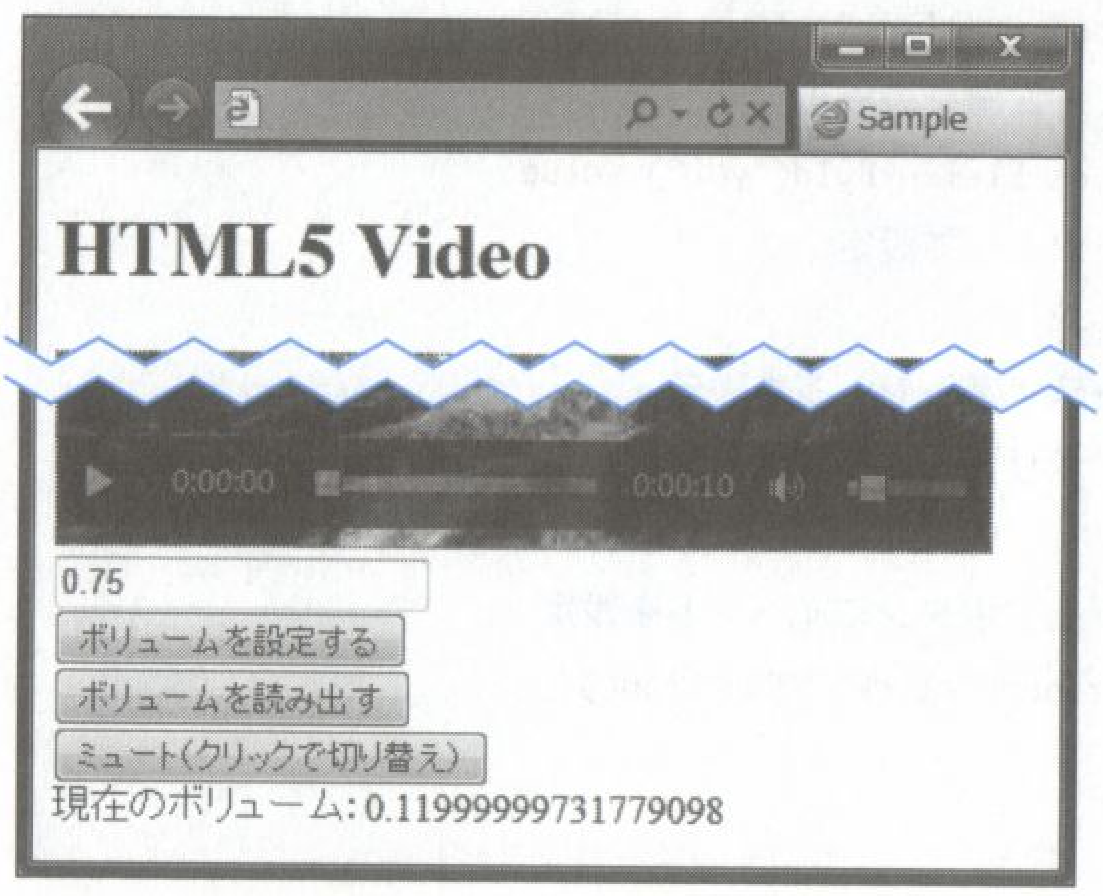
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ビデオオブジェクトを生成する
  var videoObj = document.getElementsByTagName("video")[0];
  // イベントを割り当てるボタン要素を読み出し
  var btn = document.getElementsByTagName("input");
  // ボリューム設定を行うボタンにイベントを設定
  btn[1].addEventListener("click", function(){
    // 設定するボリューム値を読み出し
    var n = document.getElementById("vol").value;
    // 映像のボリュームとして設定
    videoObj.volume = n;
    // 設定したことを示すメッセージを表示
    ele.innerHTML = "ボリュームを設定しました";
  }, false);
  // ボリュームを読み出すボタンにイベントを設定
  btn[2].addEventListener("click", function(){
```

```
      // 現在のボリューム値を読み出し
      var n = videoObj.volume;
      // 設定したことを示すメッセージを表示
      ele.innerHTML = "現在のボリューム :"+n;
    }, false);
    // ミュートを設定するボタンにイベントを設定
    btn[3].addEventListener("click", function(){
      // 現在のミュート状態を読み出し値を反転する
      var flag = videoObj.muted = !videoObj.muted;
      // 設定したことを示すメッセージを表示
      ele.innerHTML = "ミュート状態 :"+flag;
    }, false);
  }, false);
```

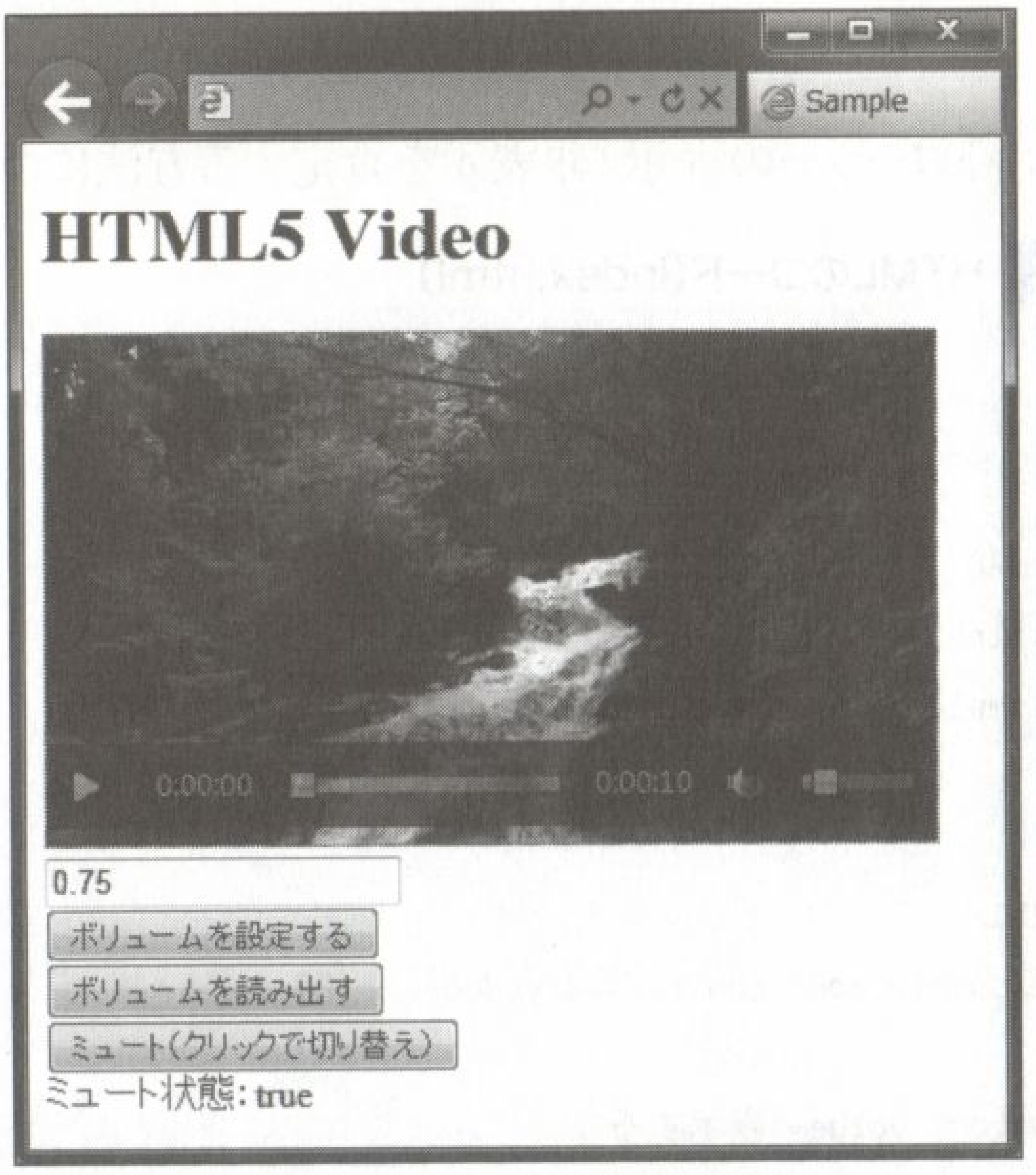
ブラウザでindex.htmlを表示し、テキストフィールドに設定したいボリューム値(「0.0」～「1.0」)を入力後、「ボリュームを設定する」ボタンをクリックすると、ボリュームが設定されます。



「ボリュームを読み出す」ボタンをクリックすると、現在のボリューム値が表示されます。



「ミュート(クリックで切り替え)」ボタンはクリックするたびに、ミュートのON/OFFが切り替わります。



ONEPOINT

ボリュームを調整するには「volume」プロパティ、ミュートを設定するには「muted」プロパティを使う

HTML5 Videoでボリューム(音量)を設定するには、ビデオオブジェクトの「volume」プロパティに値を代入します。設定できる値は「0.0」〜「1.0」の範囲になります。この範囲を超えた場合、自動的に値を調整して設定される場合もあります(負数の場合は設定されない場合がある)。また、現在のボリュームを読み出すには、ビデオオブジェクトの「volume」プロパティを参照します。

ボリュームの調整ではなく、一時的に音を消す(ミュート/消音)場合には、ビデオオブジェクトの「muted」プロパティに「true」を設定します。再度音を出すようにするには「false」を設定します。

なお、ボリューム関係は、スマートフォンでは読み出すことはできても設定することができない場合があります。その際、設定してもエラーは発生しません。

関連項目 ▶▶▶

● 映像のボリュームが変更された際のイベントを取得する……………………p.665

SECTION-279

映像のコントローラーの表示/非表示を設定する

ここでは、映像のコントローラーの表示/非表示を設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="表示する">
      <input type="button" value="非表示にする">
    </form>
  </body>
</html>
```

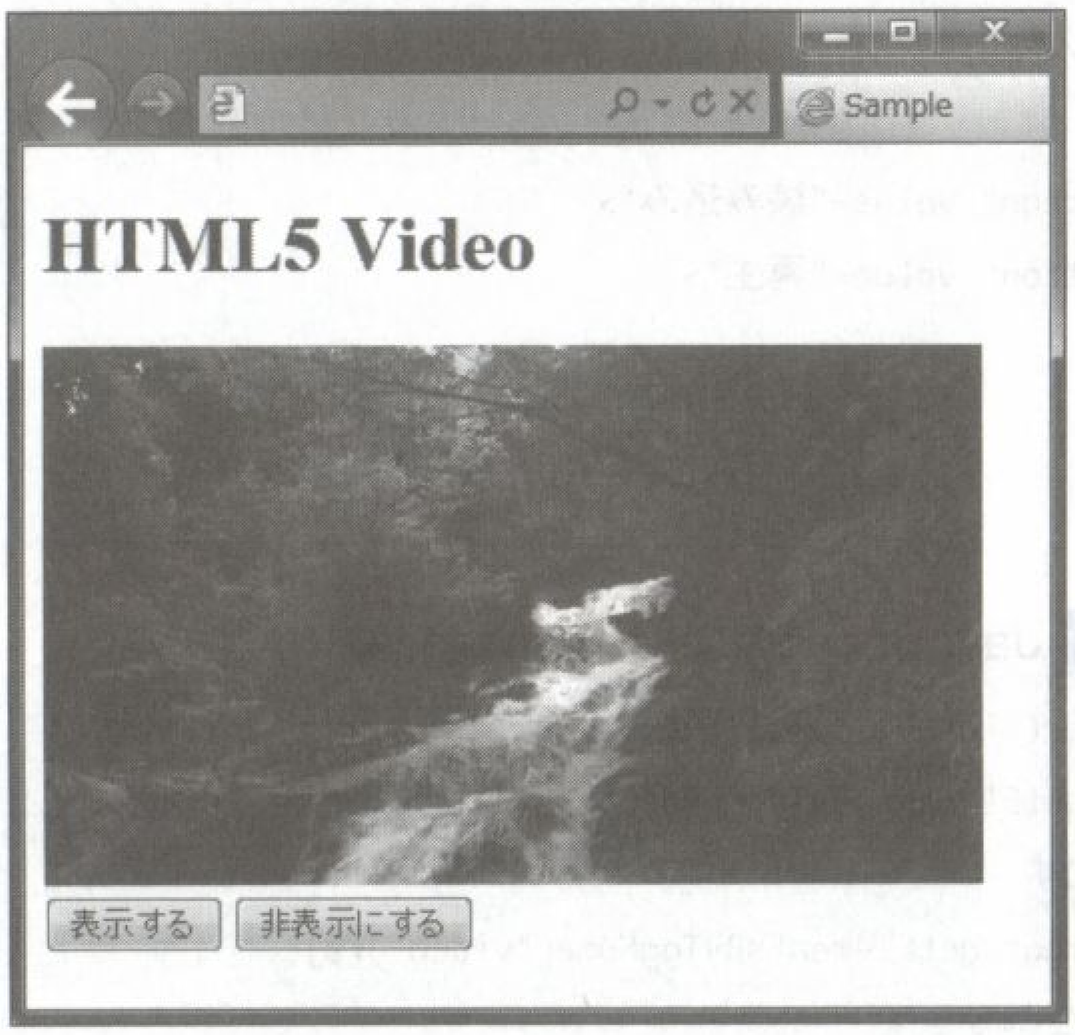
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // ビデオオブジェクトを読み出し
  var videoObj = document.getElementsByTagName("video")[0];
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // コントローラーを表示する
    videoObj.controls = true;
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // コントローラーを非表示にする
    videoObj.controls = false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、最初の状態ではコントローラーは表示されています。



「非表示にする」ボタンをクリックすると、コントローラーが消えます。



ONEPOINT　コントローラーの処理を行うには「controls」プロパティを使う

「video」要素では、コントローラーを表示するための「controls」属性が用意されています。スマートフォンやタブレットによっては、「controls」属性を指定しないと再生することができない（表示されない）場合があります。また、コントローラーは表示する必要があるが、再生中は消しておきたい場合もあります。

コントローラーの表示制御は、ビデオオブジェクトの「controls」プロパティに「true」、または、「false」を代入することで行います。「true」にするとコントローラーが表示され、「false」にするとコントローラーは非表示になります。

なお、コントローラーの表示サイズや、表示されるボタンなどはブラウザやバージョンによって異なります。また、iPhone/Android 2/Windows Phoneではページ内で映像を再生することができないため、コントローラーの表示処理は実質的に行われません。

映像データを読み込む

ここでは、後から映像データを読み込む方法について解説します。

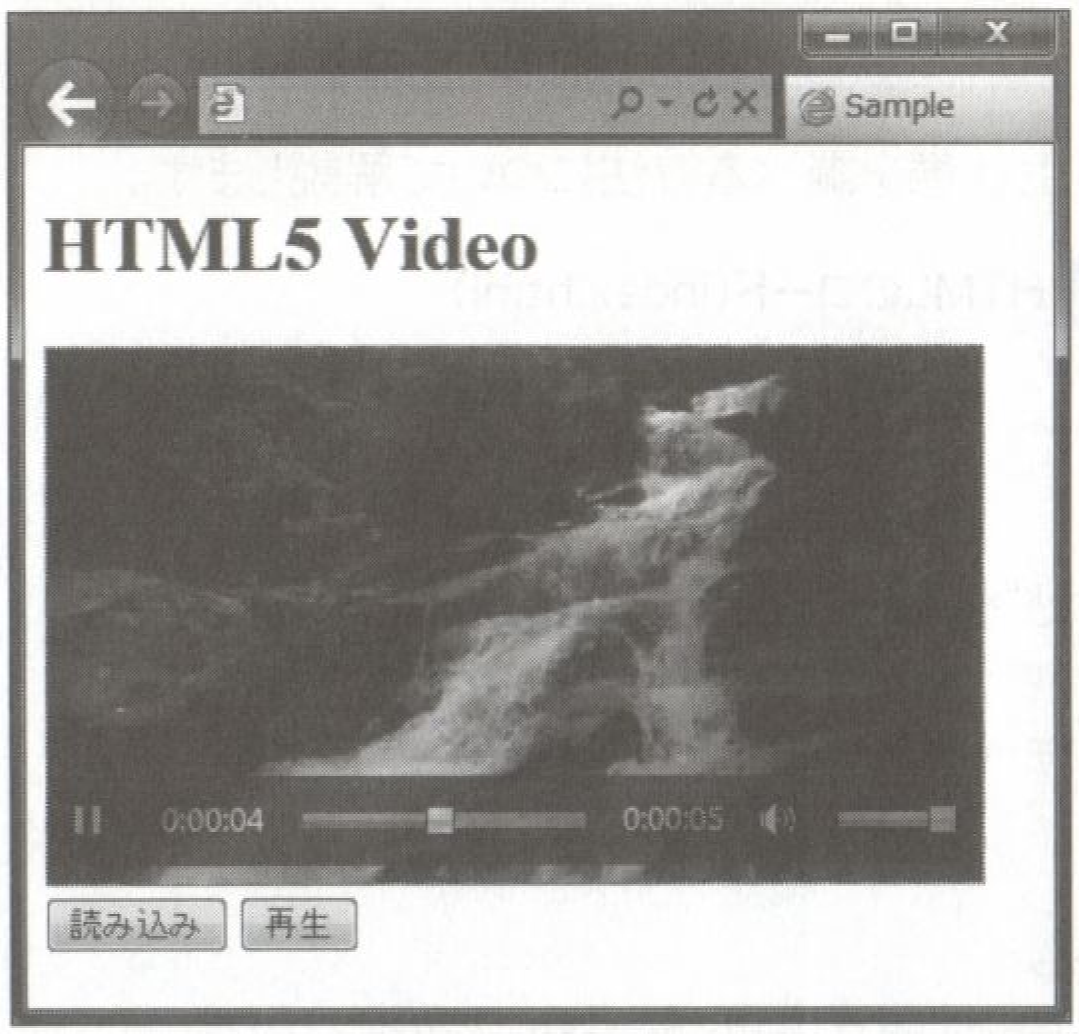
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <form>
      <input type="button" value="読み込み">
      <input type="button" value="再生">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // video要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // 映像ファイルを読み込み
    videoObj.src = "movie/matsumotojyo.mp4";
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // 映像を再生する
    videoObj.play();
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、再生ボタンをクリックすると、最初に読み込まれた映像が再生されます。



再生途中で読み込みボタンをクリックすると、他の映像ファイルが読み込まれ画面が切り替わります。再生ボタンをクリックすると、最初から映像が再生されます。



ONEPOINT　映像データを読み込ませるには「src」プロパティを使う

映像ファイルを状況に合わせて変えて再生したい場合があります。このような場合、ビデオオブジェクトの「src」プロパティに再生したい映像ファイルのURLを指定します。これにより、動的に再生される映像を変えることができます。

関連項目 ▶▶▶

● 画像や映像ファイルを読み込む ………………………………………………… p.687

SECTION-281

映像の再生状態を調べる

ここでは、映像の再生状態を調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls loop></video>
    <br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // video要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // タイマーを使って映像の状態を監視する
  setTimeout(function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生速度を表示
    ele.innerHTML = "playbackRate:"+videoObj.playbackRate;
    // 一時停止状態を表示
    ele.innerHTML += "<br>paused:"+videoObj.paused;
    // 再生終了状態を表示
    ele.innerHTML += "<br>ended:"+videoObj.ended;
    // 自動再生状態を表示
    ele.innerHTML += "<br>autoplay:"+videoObj.autoplay;
    // ループ(繰り返し再生)状態を表示
    ele.innerHTML += "<br>loop:"+videoObj.loop;
    // タイマー処理を継続
    setTimeout(arguments.callee, 250);
  }, 250);
}, false);
```

ブラウザでindex.htmlを表示すると、映像の状態がコントローラーの下に表示されます。



映像を再生すると、プロパティの状態が変化します。



ONEPOINT **映像の状態を取得するには「currentTime」プロパティや「paused」プロパティなどを使う**

HTML5 Videoでは、さまざまなプロパティがあります。その中で再生状態を示すプロパティは、次の表のようになります。プロパティは、どのブラウザでも読み出すことができます。

プロパティ	説明
currentTime	現在の再生時間(再生ヘッドの位置)
duration	映像の長さ
paused	停止しているかどうか。停止していれば「true」、再生時は「false」
playbackRate	再生速度(等倍速だと1.0、2倍速なら2.0)
ended	再生が終了しているか。終了したら「true」、そうでなければ「false」
autoplay	自動再生するかどうか。自動再生する場合は「true」、そうでない場合は「false」
loop	繰り返し(ループ)再生するかどうか。「true」なら繰り返し、「false」なら繰り返さない。ただし、Firefoxではloopが機能しない

SECTION-282

映像の再生速度を変更する

ここでは、映像の再生速度を変更する方法について解説します。

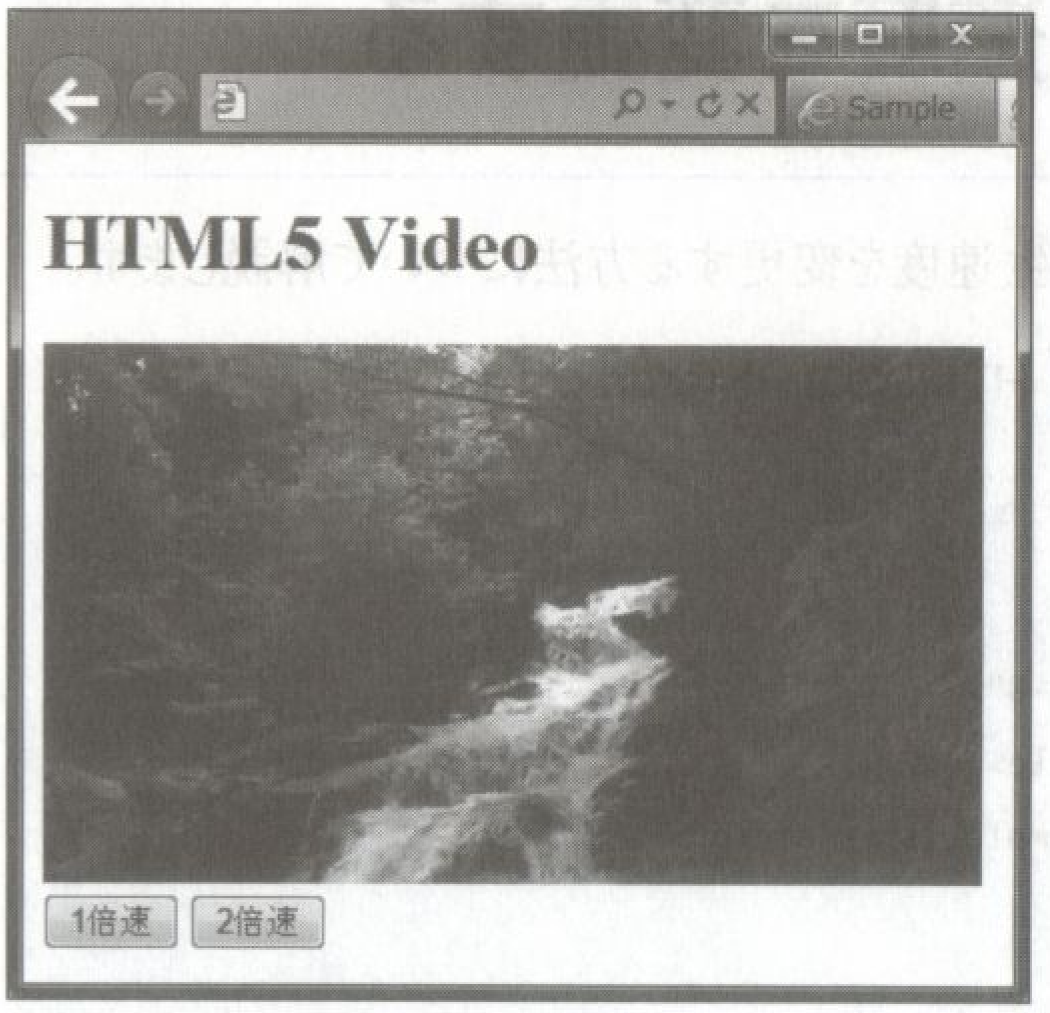
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls loop></video>
    <form>
      <input type="button" value="1倍速">
      <input type="button" value="2倍速">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var btn = document.getElementsByTagName("input");
  // video要素を読み出し
  var videoObj = document.getElementsByTagName("video")[0];
  // ボタンにイベントを割り当て
  btn[0].addEventListener("click", function(){
    // 1倍速で再生する
    videoObj.playbackRate = 1.0;
    videoObj.defaultPlaybackRate = 1.0;
  }, false);
  // ボタンにイベントを割り当て
  btn[1].addEventListener("click", function(){
    // 2倍速で再生する
    videoObj.playbackRate = 2.0;
    videoObj.defaultPlaybackRate = 2.0;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、再生速度が変更されます。



ONEPOINT 再生速度を変更するには「playbackRate」プロパティを使う

IE9以降やGoogle Chrome、Safariでは、映像の再生速度を変更できます。この場合、ビデオオブジェクトの「playbackRate」プロパティに再生速度の倍率を指定します。「1.0」であれば等倍速に、「0.5」なら半分の速度（スロー再生）、「2.0」なら倍速再生になります。あまり、大きい数値や小さい数値を指定すると、指定した通りの速度で再生されないことがあります。IE9の場合、「playbackRate」プロパティに値を設定してから実際に反映されるまでに数秒かかることもあります。また、IE9では映像がループ再生されている場合、映像の最初に戻ると「playbackRate」プロパティに設定した値は反映されずデフォルト値に戻ります。

デフォルトの再生速度を設定する場合には、「defaultPlaybackRate」プロパティに再生速度の倍率を指定します。「defaultPlaybackRate」プロパティを設定しておけば、IE9でも、ループ再生で最初に戻ったときに設定した速度で映像が再生されます。

SECTION-283

映像の再生時のイベントを取得する

ここでは、映像の再生時のイベントを取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
    <br>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // video要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // 映像にイベントリスナーを設定
  videoObj.addEventListener("timeupdate", function(){
    var ele = document.getElementsByTagName("output")[0];
    // 再生時間を表示
    ele.innerHTML = "再生時間 :"+this.currentTime;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、映像を再生すると、コントローラーの下に再生時間が表示されます。また、再生していない状態でも、コントローラー上のスライダーをドラッグすると、再生時間が表示されます。



ONEPOINT 映像の再生時の処理を設定するには「timeupdate」イベントを使う

HTML5 Videoでは、現在の再生時間を示す「currentTime」プロパティの値が変化した場合に、「timeupdate」イベントが発生します。「play()」メソッドを使ったり、コントローラーで再生したりした際に発生するのはもちろん、コントローラーで再生時間を示すスライダーをドラッグした場合にも「timeupdate」イベントは発生します。

「timeupdate」イベントの発生間隔は、ブラウザや動作環境(OSやバージョン)によって異なります。Firefoxのように頻繁に発生するブラウザもあれば、Safariのように1秒間に数回しか発生しないブラウザもあります。

より短い間隔で再生時間を取得したい場合には、「setTimeout()」メソッドか、「setInterval()」メソッドなどのタイマー処理行うメソッドを使う必要があります。

関連項目 ▶▶▶

- イベントハンドラを設定する ……… p.396
- イベントリスナーを設定する ……… p.398
- 映像を再生・停止する ……… p.639
- 映像の再生が終了した際のイベントを取得する ……… p.663
- 映像のボリュームが変更された際のイベントを取得する ……… p.665

SECTION-284

映像の再生が終了した際のイベントを取得する

ここでは、映像の再生が終了した際のイベントを取得する方法について解説します。

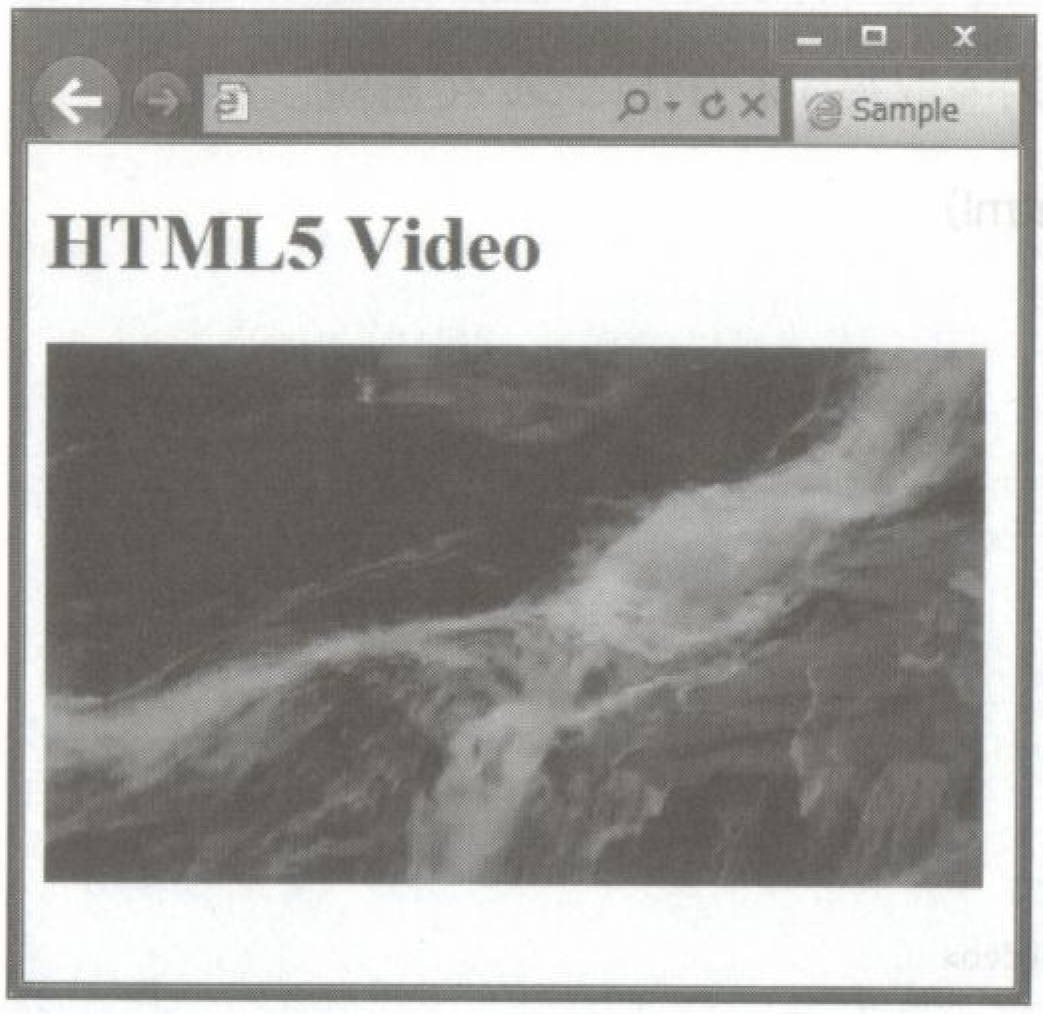
SAMPLE CODE HTMLのコード(index.html)

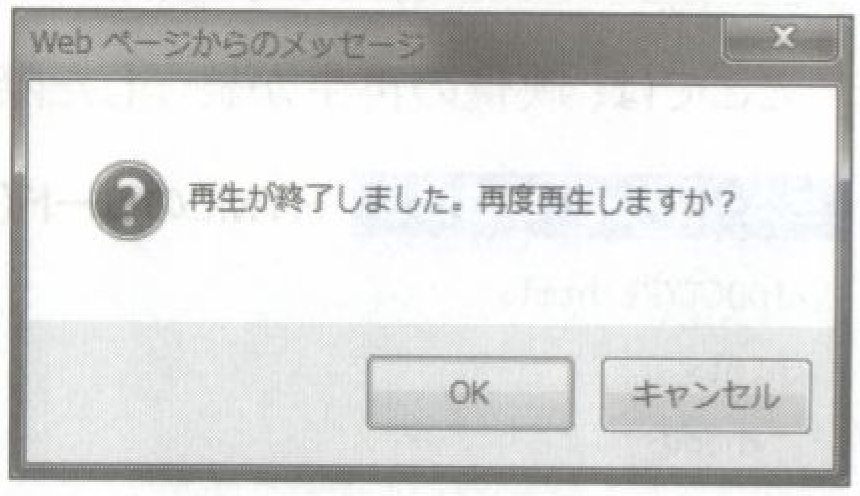
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls></video>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // video要素を読み出す
  var videoObj = document.getElementsByTagName("video")[0];
  // 映像にイベントリスナーを設定
  videoObj.addEventListener("ended", function(){
    var flag = confirm("再生が終了しました。再度再生しますか？ ");
    if (flag === true){
      // 再生時間を0(最初)に設定する
      this.currentTime = 0;
      // 再生する
      this.play();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、オーディオの再生が終了すると再生を再度、行うかどうかのダイアログが表示されます。「はい」ボタンをクリックすると再度、映像が再生され、「いいえ」ボタンをクリックするとそこで再生は終了して元の画面に戻ります。





ONEPOINT 映像の再生が終了したら処理するには「ended」イベントを使う

HTML5 Videoでは、再生が終了すると「ended」イベントが発生します。ゲームのようにオープニングで映像の再生が終了したら他のページに移動したり、次の処理を行ったりする場合に利用できます。ただし、映像で繰り返し再生を行う「loop」属性が設定されていると、「ended」イベントは発生しません。なお、Firefoxでは「loop」属性を設定しても無効なので、常に「ended」イベントが発生することになります。Firefoxで繰り返し再生する場合は、「ended」イベントを検知したら「currentTime」プロパティに「0」を設定し、再度、「play()」メソッドを使って最初から再生する必要があります。

Opera 12では、ユーザーがコントローラーを操作して再生時間を変更した場合、「ended」イベントが2回発生します。

関連項目 ▶▶▶

- イベントハンドラを設定する ……… p.396
- イベントリスナーを設定する ……… p.398
- 映像を再生・停止する ……… p.639
- 映像の再生時のイベントを取得する ……… p.661
- 映像のボリュームが変更された際のイベントを取得する ……… p.665

SECTION-285

映像のボリュームが変更された際のイベントを取得する

ここでは、ビデオのボリュームが変更された際のイベントを収録する方法について解説します。

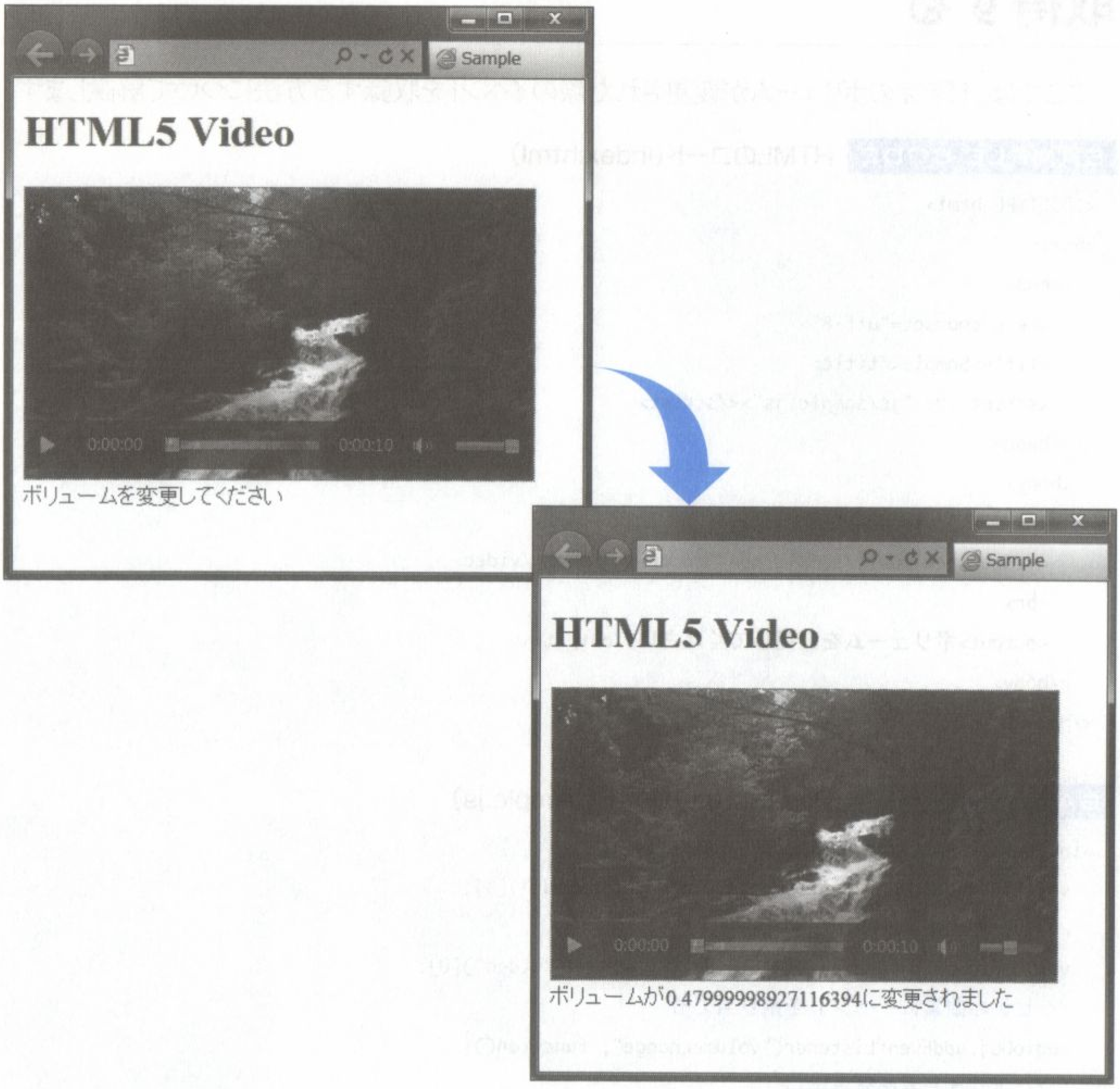
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>HTML5 Video</h1>
    <video src="movie/sample.mp4" controls loop></video>
    <br>
    <output>ボリュームを変更してください</output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ビデオ要素を読み出し
  var audioObj = document.getElementsByTagName("video")[0];
  // ビデオ要素にイベントを割り当てる
  audioObj.addEventListener("volumechange", function(){
    // ボリュームを読み出し
    var n = audioObj.volume;
    // ボリュームが変更されたことを示すメッセージを表示
    ele.innerHTML = "ボリュームが"+n+"に変更されました";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、コントローラーのボリュームスライダーをドラッグすると、リアルタイムに変更されたボリュームの値が表示されます。



ONEPOINT **ボリューム変更時の処理を行うには「volumechange」イベントを使う**

HTML5 Videoでボリューム（音量）が変更されると、「volumechange」イベントが発生します。「volumechange」イベントが発生した後のボリュームの値は、変更後のものになります。ボリュームの値は「0.0」～「1.0」までの範囲の小数値になります。

なお、スマートフォン（iPhone/iPad/Android）では本体のボリュームを変更しても、「volumechange」イベントは発生しません。

関連項目 ▶▶▶

● イベントハンドラを設定する …… p.396
● イベントリスナーを設定する …… p.398
● 映像のボリュームを設定・ミュートする …… p.649
● 映像の再生時のイベントを取得する …… p.661
● 映像の再生が終了した際のイベントを取得する …… p.663

SECTION-286

カメラからの映像を表示する

ここでは、WebRTCを使ってカメラ映像を表示する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>カメラ映像を表示</h3>
    <video width="320" height="240" autoplay></video>
    <output></output>
  </body>
</html>
```

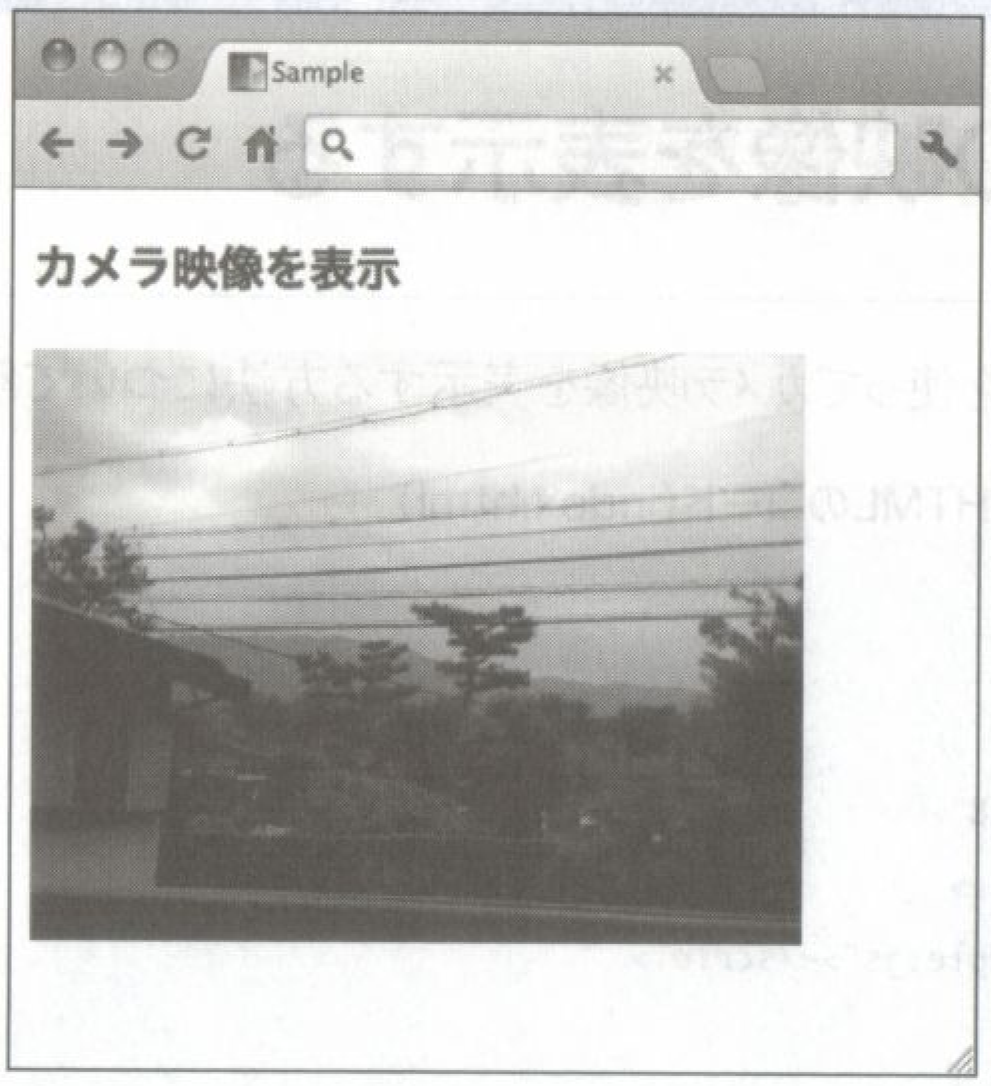
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 映像と音声の両方を再生。
  navigator.webkitGetUserMedia({ video : true, audio: true },
  // 処理できる場合
    function(stream){
      // ストリーム映像を表示するvideo要素
      var myVideo = document.getElementsByTagName("video")[0];
      // video要素のsrcプロパティにストリームをURLに変換した後に代入
      // Firefoxの場合はwindow.URL.createObjectURL()
      myVideo.src = webkitURL.createObjectURL(stream);
    },
    // エラーの場合
    function(error){
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML = "エラー:"+error.code;
    }
  );
}, false);
```

HINT

サンプルの動作は、ネットワーク環境(Webサーバー上)で確認してください。また、Oprera Mobileでの動作については、669ページのコラムを参照してください。

ブラウザでindex.htmlを表示し、ページが表示されると、カメラからの映像が取り込まれて表示されます。



ONEPOINT カメラ映像を表示するにはWebRTC APIを使う

カメラ映像を表示するには、WebRTC APIを使います。カメラ映像を取得するには、「navigator」オブジェクトの「getUserMedia()」メソッドを使います。「getUserMedia()」メソッドの最初のパラメータは、取得するデバイスの種類を指定します。古い仕様では、「"video, audio"」として文字列で指定します。新しい仕様では、「{ video:true, audio:false }」としてオブジェクトリテラル形式で指定します。

「getUserMedia()」の2番目のパラメータには、デバイスからの入力に成功した際に呼び出す関数を指定します。3番目のパラメータには、失敗したときに呼び出す関数を指定します。

デバイスからの入力に成功した場合、関数にはストリームデータが渡されます。ただし、このままでは「video」要素に映像を表示することはできません。映像として表示するには、「createObjectURL()」メソッドを利用します。「createObjectURL()」メソッドでURL形式に変換した後、「video」要素の「src」プロパティに代入することでカメラ映像が表示されます。ただし、「video」要素に「autoplay」属性(「autoplay="autoplay"」)が指定されていない場合は、最初に取り込まれた1フレームの映像だけが表示されます。「autoplay」を指定することで、リアルタイムにカメラ映像を表示することができます。

なお、サンプルではベンダープレフィックス(webkit)を付加しています。

COLUMN Google Chromeでの設定

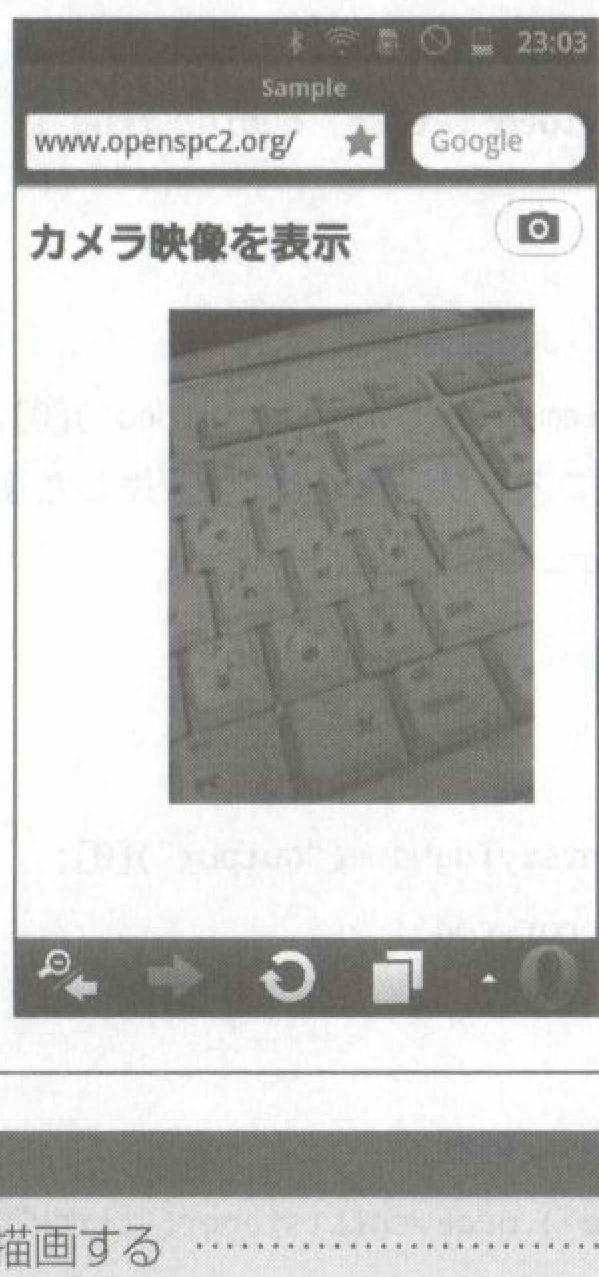
「getUserMedia()」メソッド(実際には「navigator.webkitGetUserMedia()」メソッド)が存在しない場合は、アドレス欄に「about:flags」と入力して、設定画面で「Media Stream」を有効にしてください。設定後、再起動すると、「getuserMedia()」メソッドを利用することができるようになります。

COLUMN　Opera MobileでのWebRTC

WebRTCは、Opera Mobileでも利用することができます。Opera Mobileの場合、「navigator」オブジェクトの「getUserMedia()」メソッドを使ってリアルタイムに映像を表示することができます。実際のJavaScriptのコードは、次のようになります(HTMLについては、サンプルと同じ)。

```
window.addEventListener("load", function(){
  // 映像と音声の両方を再生
  navigator.getUserMedia("video, audio",
    // 処理できる場合
    function(stream){
      // ストリーム映像を表示するvideo要素
      var myVideo = document.getElementsByTagName("video")[0];
      // Opera Mobileではvideo要素のsrcプロパティにストリームを入れる
      myVideo.src = stream;
    },
    // エラーの場合
    function(error){
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML = "エラー:"+error.code;
    }
  );
}, false);
```

Opera Mobileでindex.htmlを表示すると、ページが表示されると、カメラからの映像が取り込まれて表示されます。



関連項目 ▶▶▶
●カメラからの映像をCanvasに描画する ……………………………… p.670

SECTION-287

カメラからの映像をCanvasに描画する

ここでは、WebRTCを使ってカメラ映像から取り込んだ映像をCanvasに描画する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>カメラ映像をCanvasに取り込む</h3>
    <form>
      <input type="button" value="取り込み" id="capture">
    </form>
    <video width="320" height="240" autoplay></video>
    <canvas width="320" height="240"></canvas>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

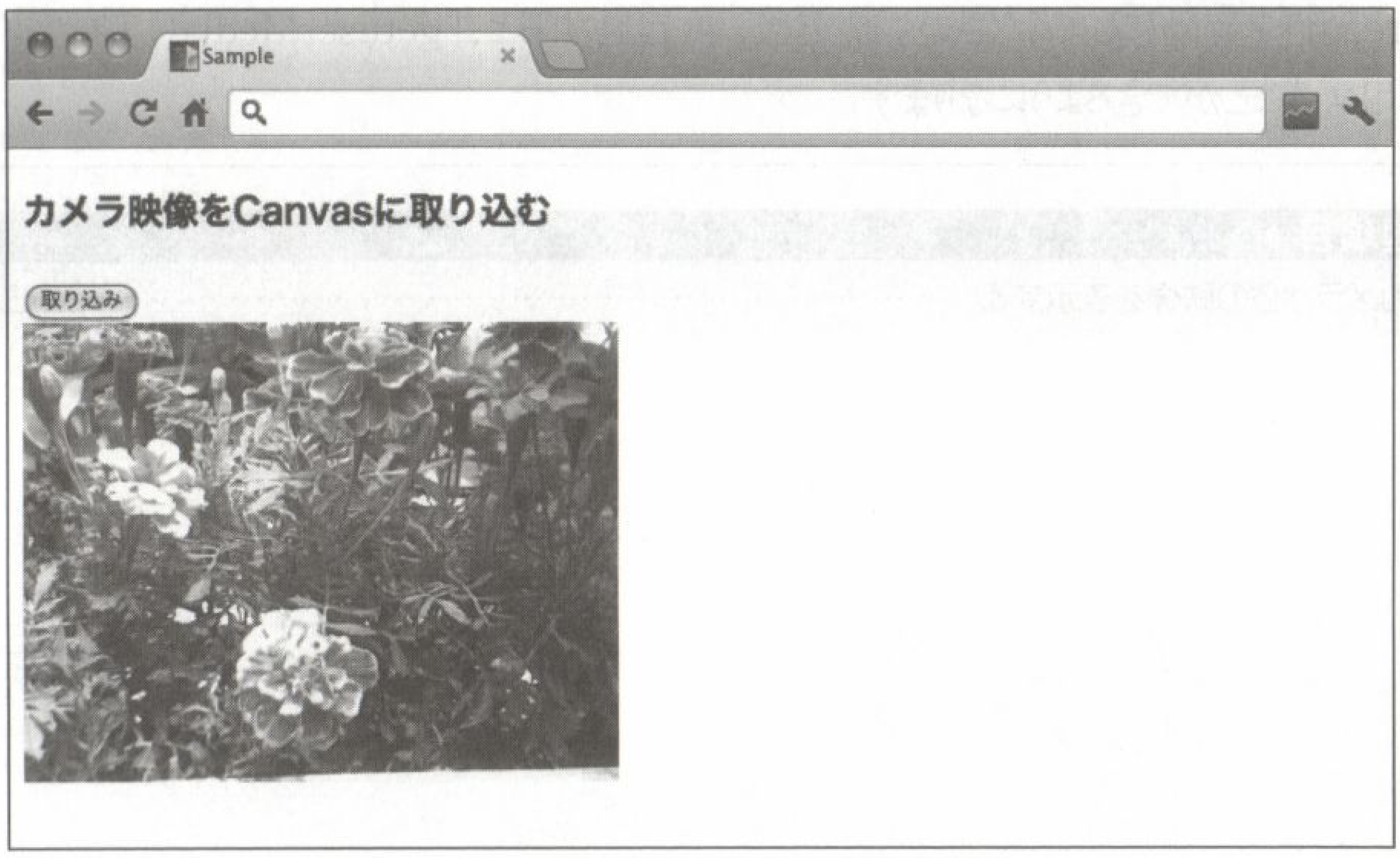
```
window.addEventListener("load", function(){
  // 映像と音声の両方を再生
  navigator.webkitGetUserMedia({ video : true, audio: true },
    // 処理できる場合
    function(stream){
      // ストリーム映像を表示するvideo要素
      var myVideo = document.getElementsByTagName("video")[0];
      // video要素のsrcプロパティにストリームをURLに変換した後に代入
      myVideo.src = webkitURL.createObjectURL(stream);
    },
    // エラーの場合
    function(error){
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML = "エラー:"+error.code;
    }
  );
  // WebRTCで取り込んだ映像をCanvasに描画
  document.getElementById("capture").addEventListener("click", function(){
    var myVideo = document.getElementsByTagName("video")[0];
```

```
    var myCanvas = document.getElementsByTagName("canvas")[0];
    var context = myCanvas.getContext("2d");
    context.fillStyle = "white";
    context.fillRect(0,0, myCanvas.width, myCanvas.height);
    context.drawImage(myVideo, 40, 40, 240, 160);
  }, true);
}, false);
```

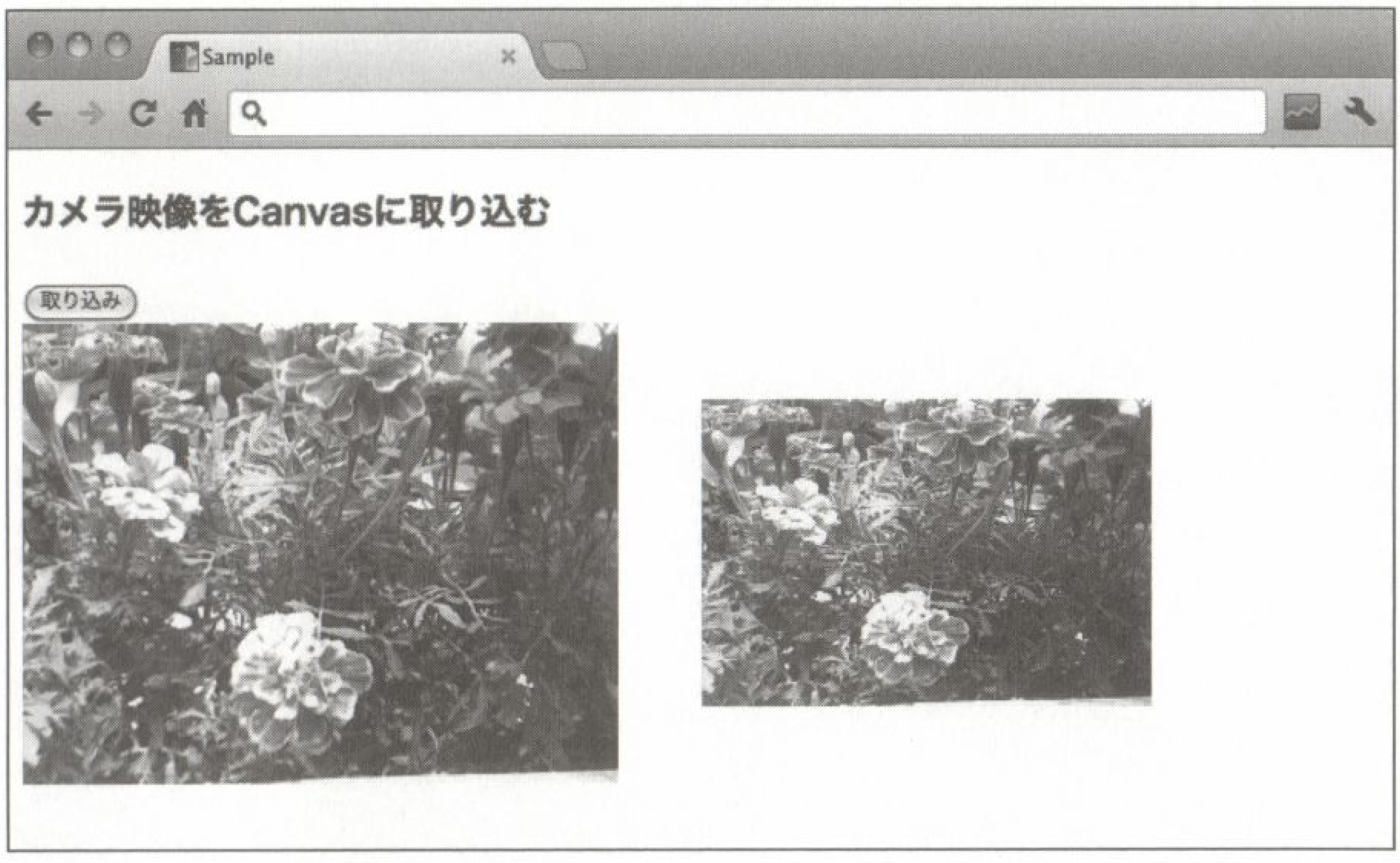
HINT

サンプルの動作は、ネットワーク環境(Webサーバー上)で確認してください。また、Oprera Mobileでの動作については、669ページのコラムを参照してください。

ブラウザでindex.htmlを表示すると、ページが表示されると、カメラからの映像が取り込まれて表示されます。



「取り込み」ボタンをクリックすると、カメラ映像がCanvasに縮小表示されます。



ONEPOINT **カメラ映像をCanvasに描画するには「drawImage()」メソッドの最初のパラメータに「video」要素を指定する**

カメラ映像をCanvasに描画するには、「drawImage()」メソッドの最初のパラメータに、映像が表示されている「video」要素を指定します。その際、「video」要素には「autoplay」属性が指定されていないと、リアルタイムに映像を描画することができません。また、映像をCanvasに描画する場合は、Canvasの内容を消去(もしくは特定色で描画)しておかないと、以前の映像が残ったまま、次の映像が描画されないことがあります。

COLUMN **Google Chromeでの設定**

「getUserMedia()」メソッド(実際には「navigator.webkitGetUserMedia()」メソッド)が存在しない場合は、アドレス欄に「about:flags」と入力して、設定画面で「Media Stream」を有効にしてください。設定後、再起動すると、「getuserMedia()」メソッドを利用することができるようになります。

関連項目 ▶▶▶

● カメラからの映像を表示する ………………………………………… p.667

SECTION-288

カメラで撮影した画像を表示する

ここでは、HTML Media Captureを使ってカメラで撮影した画像を表示する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>カメラ画像を取り込む</h3>
    <form>
      <input type="file" id="camData" accept="image/*;capture=camera">
    </form>
    <img src="images/dummy.jpg" id="myPhoto" width="400" height="320">
  </body>
</html>
```

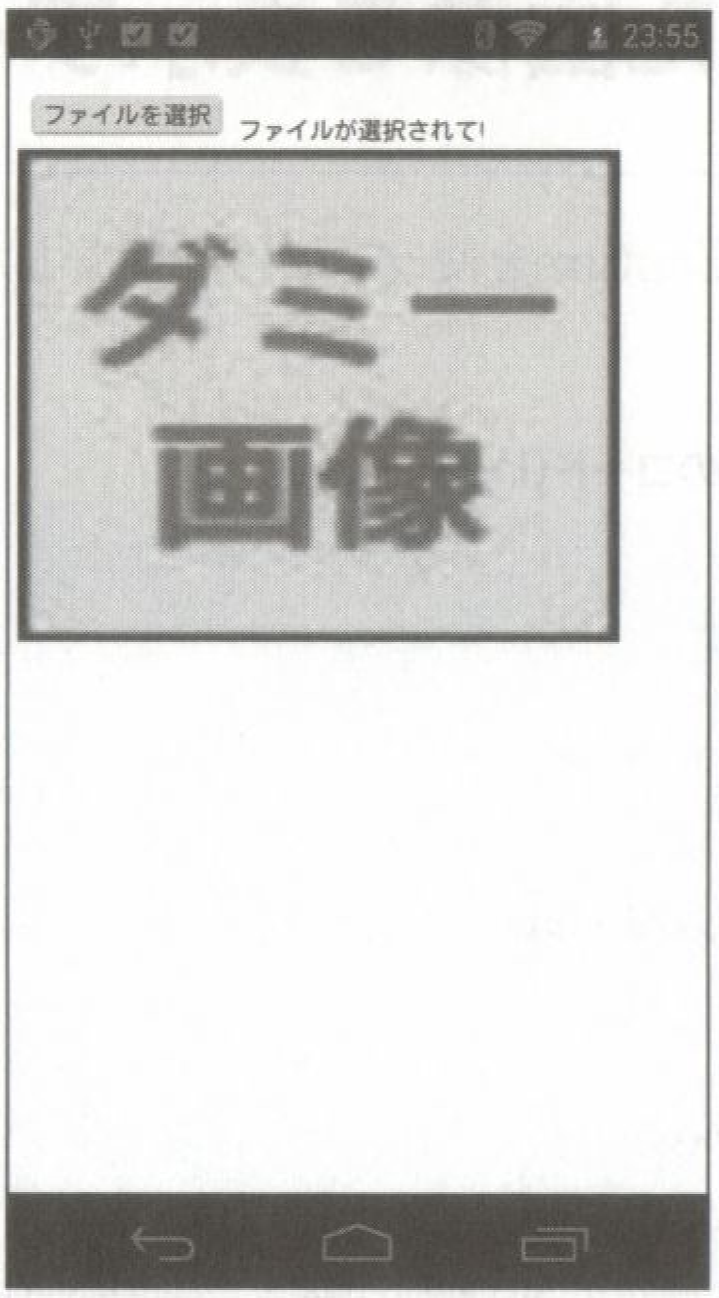
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  document.getElementById("camData").addEventListener("change", function(){
    // 撮影した画像ファイル読み込むためのオブジェクトを生成
    var reader = new FileReader();
    // 画像の読み込みが完了した時の処理
    reader.onload = function(evt){
      // カメラ画像をsrcプロパティに入れる
      document.getElementById("myPhoto").src = reader.result;
    }
    // カメラで撮影した画像データを指定して読み込ませる
    var imageFile = document.getElementById("camData").files[0];
    reader.readAsDataURL(imageFile);
  }, false);
}, false);
```

HINT

このサンプルは、スマートフォンを対象としています。

ブラウザでindex.htmlを表示し、「ファイル選択」ボタンをクリックすると、カメラが起動します。



カメラで撮影すると、撮影した画像がページ上に表示されます。



ONEPOINT　カメラで撮影した画像を表示するにはHTML Media Captureを使う

HTML Media Captureを使えば、簡単にカメラで撮影した画像を表示することができます。まず、「input」要素の「type」属性に「file」を指定し、「accept」属性に「"image/*;capture=camera"」の文字を指定します。これでファイルアップロードボタンをクリックすると、カメラでの撮影が可能になります。

カメラで撮影した画像を処理するには、File APIを利用します。「FileReader」オブジェクトを作成し、「readAsDataURL()」メソッドを使ってカメラで撮影した画像を読み込ませます。画像データの読み込みが完了したら、「img」要素の「src」属性に入れます。これでカメラで撮影した画像がページ上に表示されます。

なお、2012/7/12の仕様では、取り込みデバイスの指定は、「accept」属性ではなく、「capture」属性に指定するようになっています。

HTML Media Captureについては、次のURLを参照してください。

URL http://www.w3.org/TR/html-media-capture/

COLUMN　ビデオカメラとマイクからの取り込み

ビデオカメラから取り込むには、次のように「accept」属性を指定します。

```
<input type="file" accept="video/*;capture=camcoder">
```

マイクから取り込むには、次のように「accept」属性を指定します。

```
<input type="file" accept="audio/*;capture=microphone">
```

COLUMN　「capture」属性に指定できる値

2012/7/12時点での仕様では、「capture」属性には、次の値を指定できることになっています。

値	説明
camera	カメラ
camcorder	ビデオカメラ
microphone	マイク
filesystem	ファイル(ローカルにあるファイル)

関連項目 ▶▶▶

●カメラで撮影した画像をCanvasに描画する ……………………………… p.676

SECTION-289

カメラで撮影した画像をCanvasに描画する

ここでは、HTML Media Captureを使ってカメラで撮影した画像をCanvasに描画方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>カメラ画像を取り込む</h3>
    <form>
      <input type="file" id="camData" accept="image/*;capture=camera">
    </form>
    <canvas id="myCanvas" width="400" height="320"></canvas>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

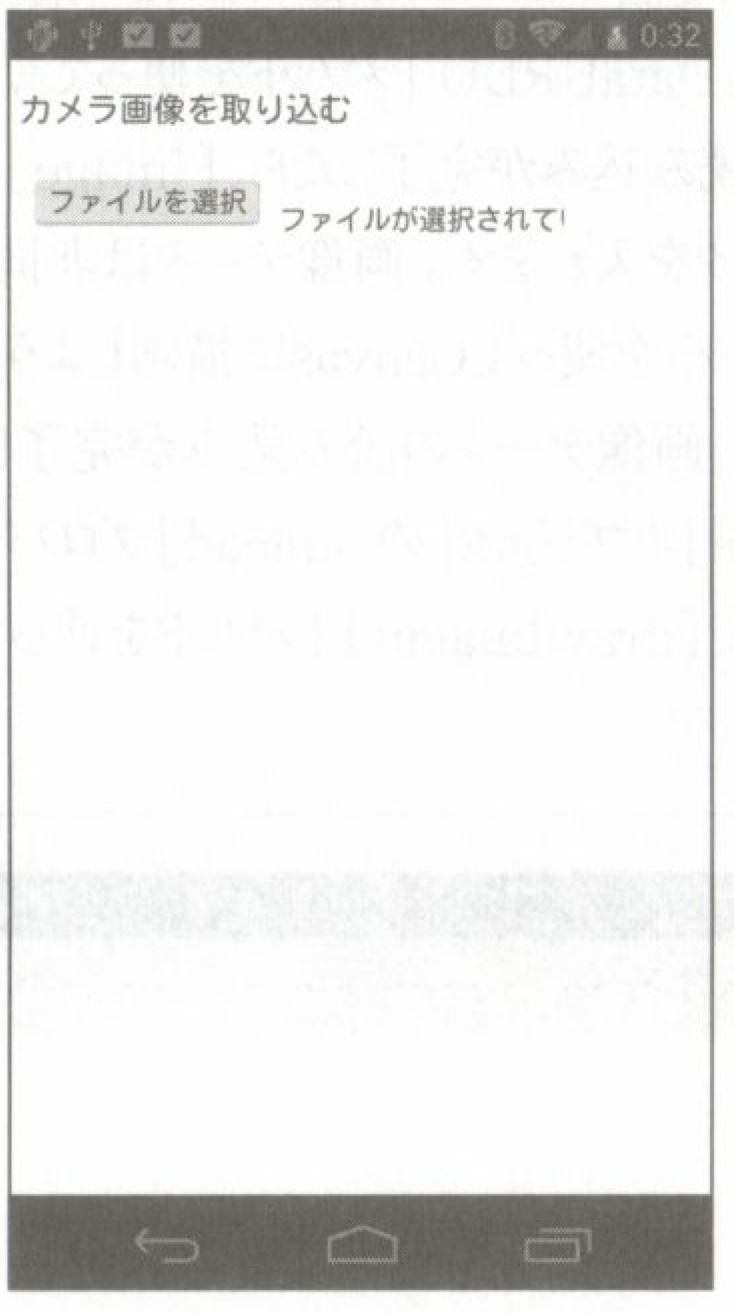
```
window.addEventListener("load", function(){
  document.getElementById("camData").addEventListener("change", function(){
    // 撮影した画像ファイル読み込むためのオブジェクトを生成
    var reader = new FileReader();
    // 画像の読み込みが完了した時の処理
    reader.onload = function(evt){
      // Imageオブジェクトを作成
      var img = new Image();
      // 画像データの読み込み完了時にCanvasに描画
      img.onload = function(){
        var canvasObj = document.getElementById("myCanvas");
        var context = canvasObj.getContext("2d");
        context.drawImage(this, 0, 0, 400, 320);
      }
      // カメラ画像をsrcプロパティに入れる
      img.src = reader.result;
    }
    // カメラで撮影した画像データを指定して読み込ませる
    var imageFile = document.getElementById("camData").files[0];
    reader.readAsDataURL(imageFile);
  }, false);
```

```
}, false);
```

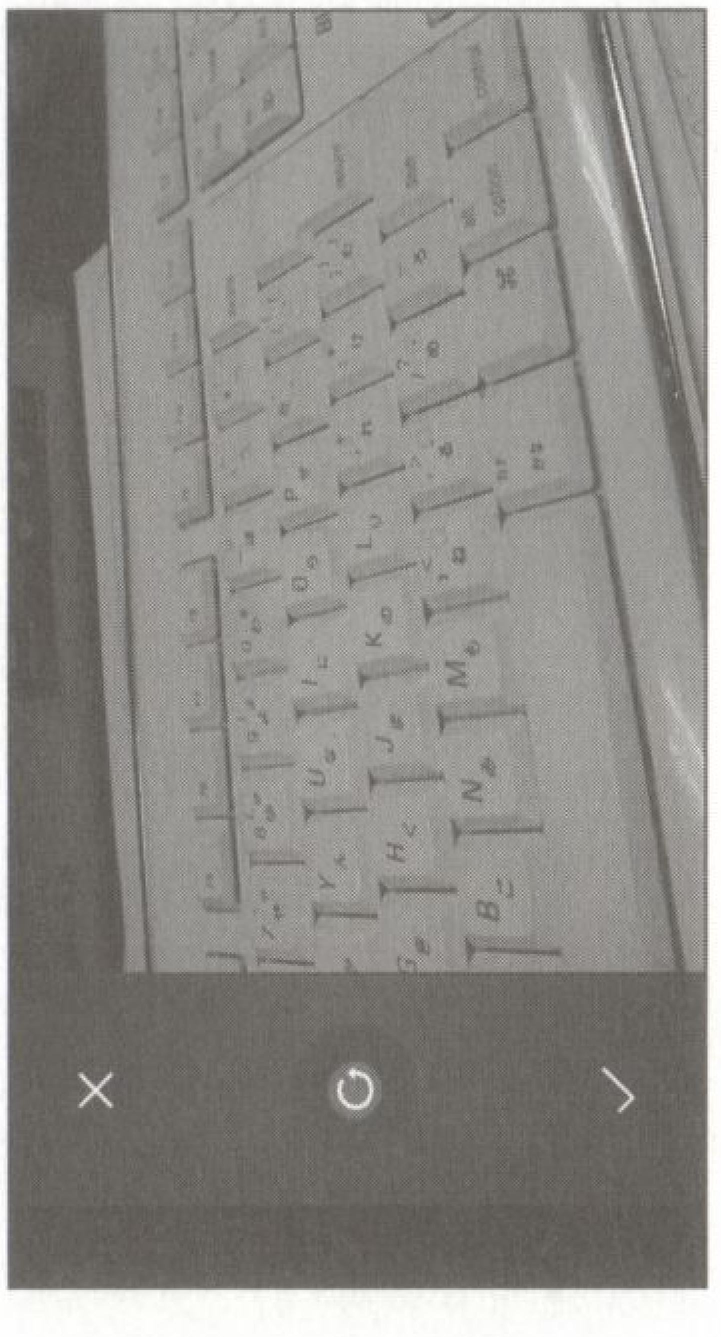
HINT

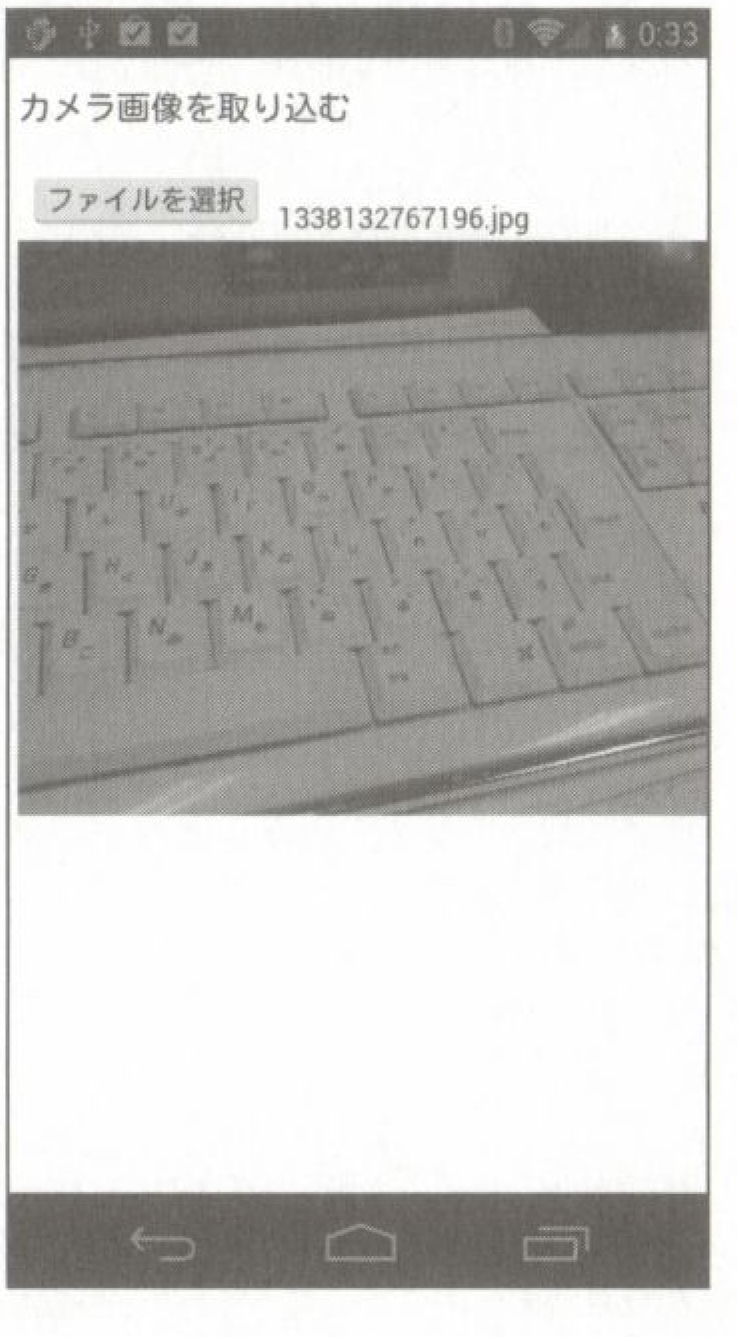
このサンプルは、スマートフォンを対象としています。

ブラウザでindex.htmlを表示し、「ファイル選択」ボタンをクリックするとカメラが起動します。



カメラで撮影すると、撮影した画像がCanvasに描画されます。





ONEPOINT **カメラで撮影した画像をCanvasに描画するには「Image」オブジェクトの「onload」プロパティにイベントハンドラを設定する**

カメラで撮影を行うには、「input」要素の「type」属性に「file」を指定し、「accept」属性に「"image/*;capture=camera"」の文字を指定します。これでファイルアップロードボタンをクリックすると、カメラでの撮影が可能になります。

カメラで撮影した画像を処理するには、File APIを利用します。「FileReader」オブジェクトを作成し、「readAsDataURL()」メソッドを使ってカメラで撮影した画像を読み込ませます。画像データの読み込みが完了したら、「Image」オブジェクトを生成し、「src」プロパティ読み込んだデータを入れます。画像データは非同期で読み込まれるため、直後に「drawImage()」メソッドを使ってCanvasに描画しようとしてもデータがないため何も描画されません。そこで、画像データの読み込みが完了したら、Canvaに描画するようにします。これは、「Image」オブジェクトの「onload」プロパティにイベントハンドラを設定します。イベントハンドラ内で「drawImage()」メソッドを使って、カメラで撮影した画像をCanvas上に描画します。

関連項目 ▶▶▶

●カメラで撮影した画像を表示する ……………………………………………… p.673

CHAPTER 12

File API

SECTION-290

ファイル名や種類を取得する

ここでは、ファイル名や種類を取得する方法について解説します。

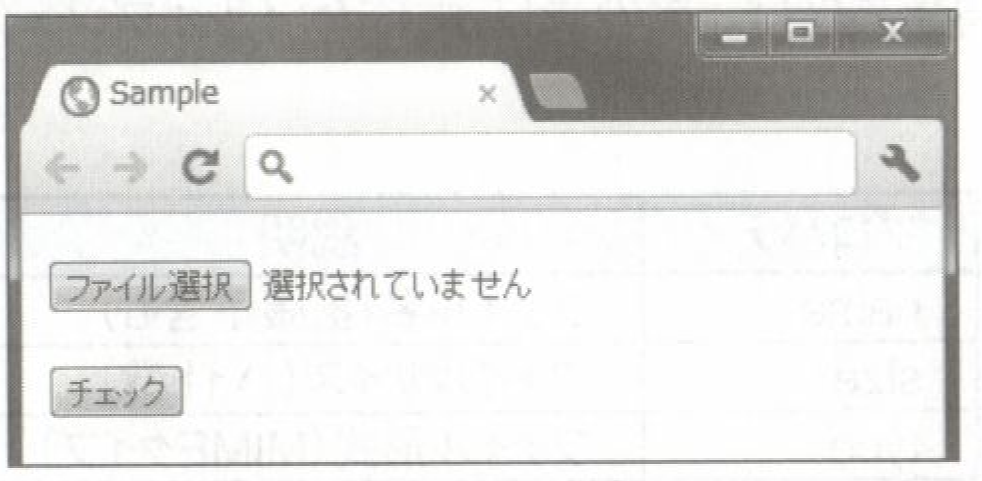
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="file" id="selFile" multiple></p>
      <input type="button" value="チェック" id="checkBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタンにイベントを設定
  document.getElementById("checkBtn").addEventListener("click", function(){
    var fileList = document.getElementById("selFile").files;
    // 選択されたファイル数
    var total = fileList.length;
    ele.innerHTML = "選択されたファイル数 :"+total+"<br>";
    // 選択されたファイル数だけ繰り返し情報を表示
    for(var i=0; i<total; i++){
      // ファイル名を読み出し
      ele.innerHTML += "ファイル名 :"+fileList[i].name+"<br>";
      // ファイルサイズを読み出し
      ele.innerHTML += "サイズ :"+fileList[i].size+"バイト<br>";
      // 種類を読み出し
      ele.innerHTML += "種類 :"+fileList[i].type+"<hr>";
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ファイル選択ボタンをクリックすると、ファイルを選択するダイアログが表示されます。



ファイルを選択します。このとき、複数のファイルを選択することができます。



チェックボタンをクリックすると、選択されたファイルの名前とサイズ、形式が表示されます。



ONEPOINT ファイル情報を取得するには「name」プロパティや「size」プロパティ、「type」プロパティを使う

ファイル情報を取得するには、次の表に示すファイルオブジェクトの各プロパティを参照します。

プロパティ	説明
name	ファイル名(拡張子含む)
size	ファイルサイズ(バイト数)
type	ファイル形式(MIMEタイプ)

ファイルオブジェクトは、ファイルアップロードボタンにより選択されるか、ページ内にドラッグ&ドロップされたファイルが「files」配列に格納されます。

ファイルオブジェクトで読み出せるのは、表に示すプロパティだけで、ファイルのパスやディレクトリなどは取得することができません。

関連項目 ▶▶▶

●ドラッグ要素からドロップ要素にデータを渡す ………… p.866

SECTION-291

テキストファイルを読み込む

ここでは、ローカルファイルにあるテキストファイルを読み込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="file" id="selFile"></p>
      <input type="button" value="読み込み" id="readBtn">
    </form>
    <output></output>
  </body>
</html>
```

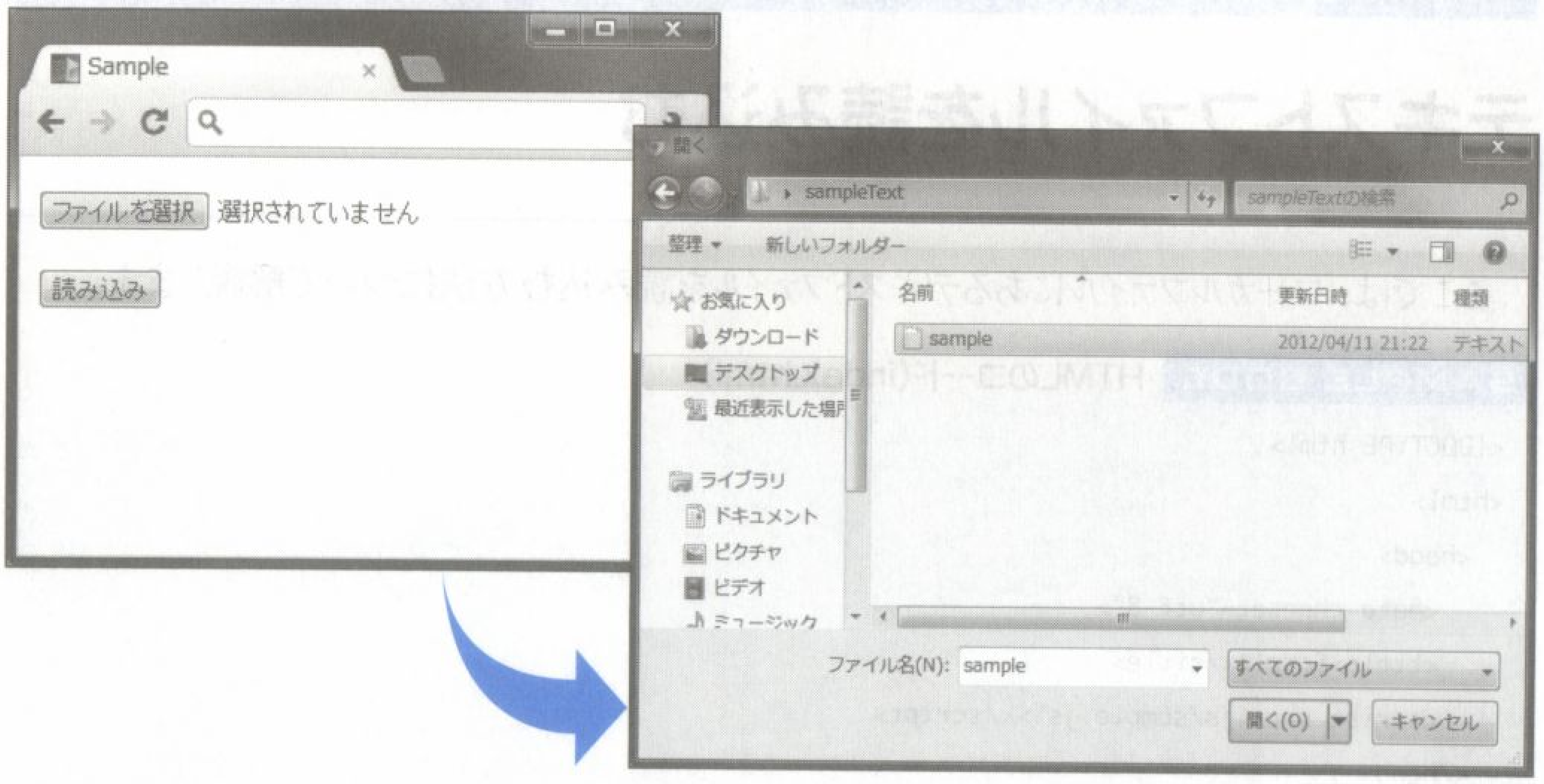
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタンにイベントを設定
  document.getElementById("readBtn").addEventListener("click", function(){
    var fileList = document.getElementById("selFile").files;
    // FileReaderオブジェクトを作成
    var reader = new FileReader();
    // テキストが読み込まれた場合の処理を設定
    reader.onload = function(evt){
      ele.innerHTML = evt.target.result;
    }
    // テキストファイルの読み込みを開始
    reader.readAsText(fileList[0]);
  }, false);
}, false);
```

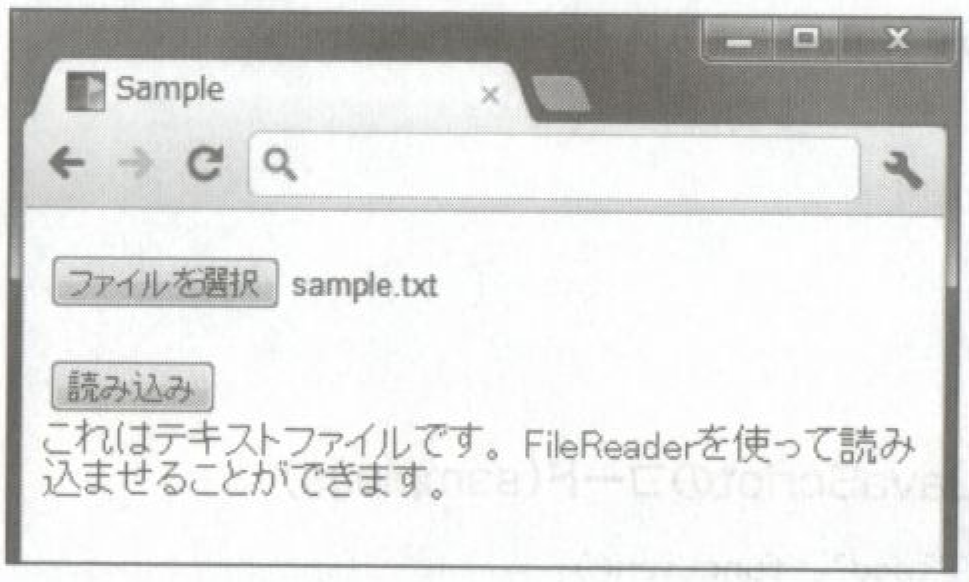
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示し、ファイル選択ボタンをクリックして、テキストファイルを選択します。



読み込みボタンをクリックすると、選択されたテキストファイルが読み込まれ、内容が表示されます。



ONEPOINT テキストファイルを読み込むには「readAsText()」メソッドを使う

テキストファイルを読み込むには、「FileReader」オブジェクトの「readAsText()」メソッドを使います。「readAsText()」のパラメータに選択されたファイルオブジェクトを指定します。「readAsText()」の2番目のパラメータは、テキストファイルの文字コードを指定します。自動的に文字コードを認識させる場合には、省略します。読み込むテキストファイルの文字コードが決まっている場合のみ、2番目のパラメータを指定する方がよいでしょう。

テキストファイルの読み込みが完了すると「load」イベントが発生するので、「FileReader」オブジェクトの「onload」にイベントハンドラを設定し、読み込み完了時の処理を行います。なお、ファイルサイズが大きい場合には、ブラウザが停止状態になることがあります。イベントハンドラにはイベントオブジェクトが渡され、「target」オブジェクトの「result」プロパティに読み込まれたテキストデータが入ります。

関連項目 ▶▶▶

- バイナリファイルを読み込む ……… p.685
- 画像や映像ファイルを読み込む ……… p.687
- ファイルの読み込み状況を表示する ……… p.689
- ドラッグ要素からドロップ要素にデータを渡す ……… p.866

SECTION-292

バイナリファイルを読み込む

ここでは、ローカルファイルにあるバイナリファイルを読み込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="file" id="selFile"></p>
      <input type="button" value="読み込み" id="readBtn">
    </form>
    <output></output>
  </body>
</html>
```

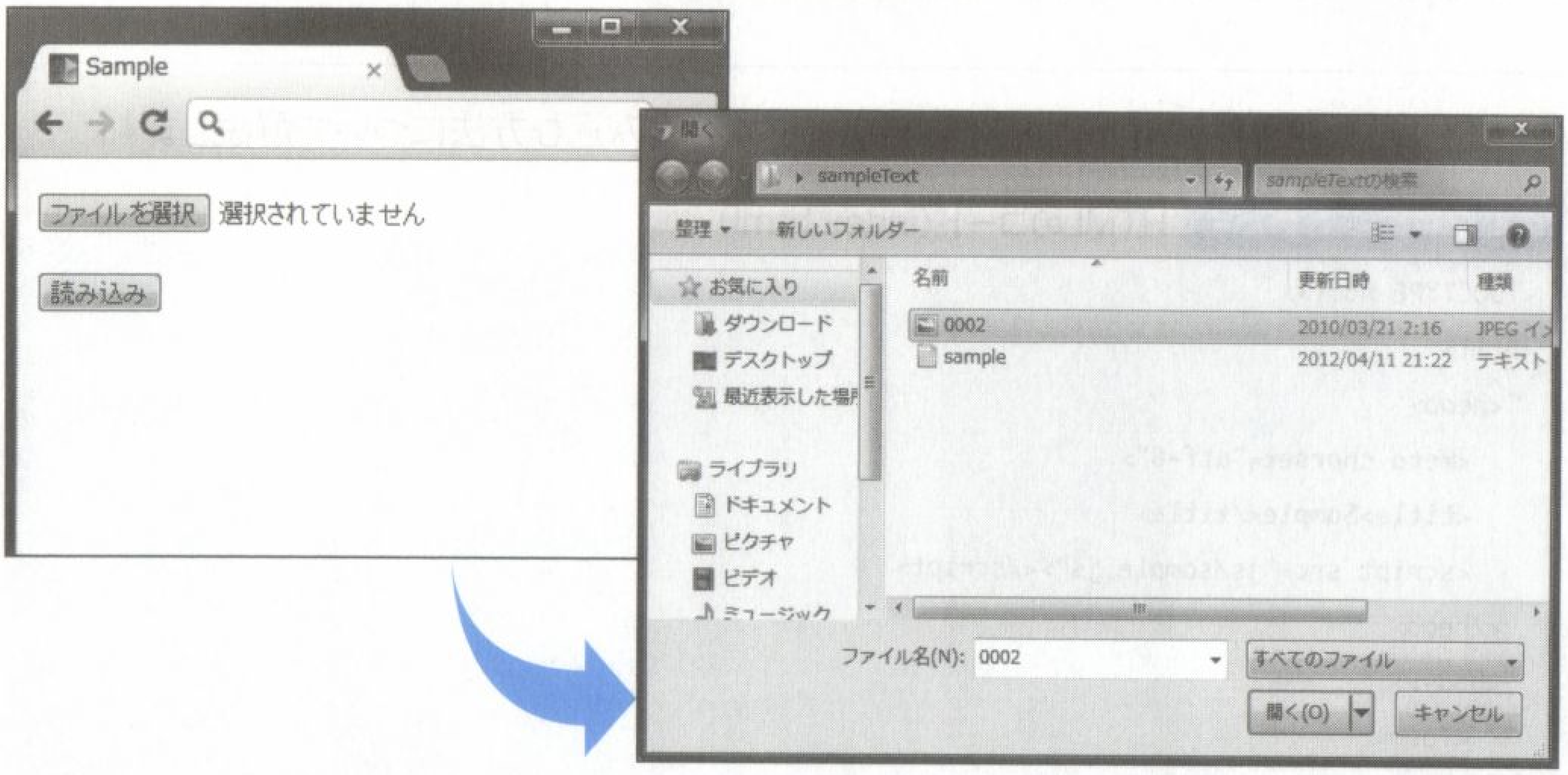
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタンにイベントを設定
  document.getElementById("readBtn").addEventListener("click", function(){
    var fileList = document.getElementById("selFile").files;
    // FileReaderオブジェクトを作成
    var reader = new FileReader();
    // バイナリファイルが読み込まれた場合の処理を設定
    reader.onload = function(evt){
      var bin = evt.target.result;
      // 最初の8バイトを表示
      for(var i=0; i<8; i++){
        ele.innerHTML += bin.charCodeAt(i).toString(16)+", ";
      }
    }
    reader.readAsBinaryString(fileList[0]);  // バイナリファイルの読み込みを開始
  }, false);
}, false);
```

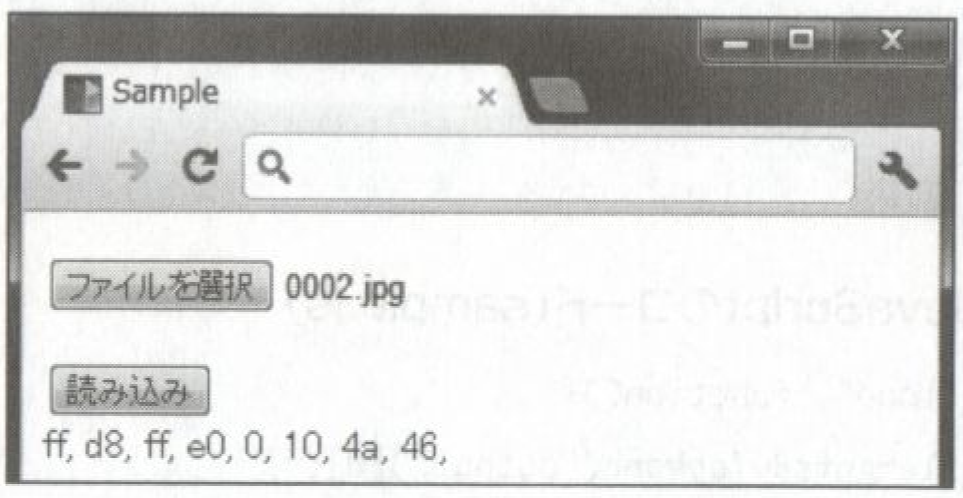
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示し、ファイル選択ボタンをクリックして、バイナリファイルを選択します。



読み込みボタンをクリックすると、選択されたバイナリファイルが読み込まれ、先頭の8バイトが16進数で表示されます。



ONEPOINT

バイナリファイルを読み込むには「readAsBinaryString()」メソッドを使う

バイナリファイルを読み込むには、「FileReader」オブジェクトの「readAsBinaryString()」メソッドを使います。「readAsBinaryString()」のパラメータに、選択されたファイルオブジェクトを指定します。

バイナリファイルの読み込みが完了すると「load」イベントが発生するので、「FileReader」オブジェクトの「onload」にイベントハンドラを設定し、読み込み完了時の処理を行います。

イベントハンドラにはイベントオブジェクトが渡され、「target」オブジェクトの「result」プロパティに読み込まれたバイナリファイルが入ります。バイナリデータは、1バイトが1文字として格納されているため、「charAt()」メソッドや「charCodeAt()」メソッド、「substring()」メソッドや「substr()」メソッド、正規表現などを使って処理することができます。

関連項目 ▶▶▶

● テキストファイルを読み込む ……… p.683
● 画像や映像ファイルを読み込む ……… p.687
● ファイルの読み込み状況を表示する ……… p.689
● ドラッグ要素からドロップ要素にデータを渡す ……… p.866

SECTION-293

画像や映像ファイルを読み込む

ここでは、ローカルファイルにある画像や映像ファイルを読み込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="file" id="selFile"></p>
      <input type="button" value="読み込み" id="readBtn">
    </form>
    <img src="images/dummy.png">
  </body>
</html>
```

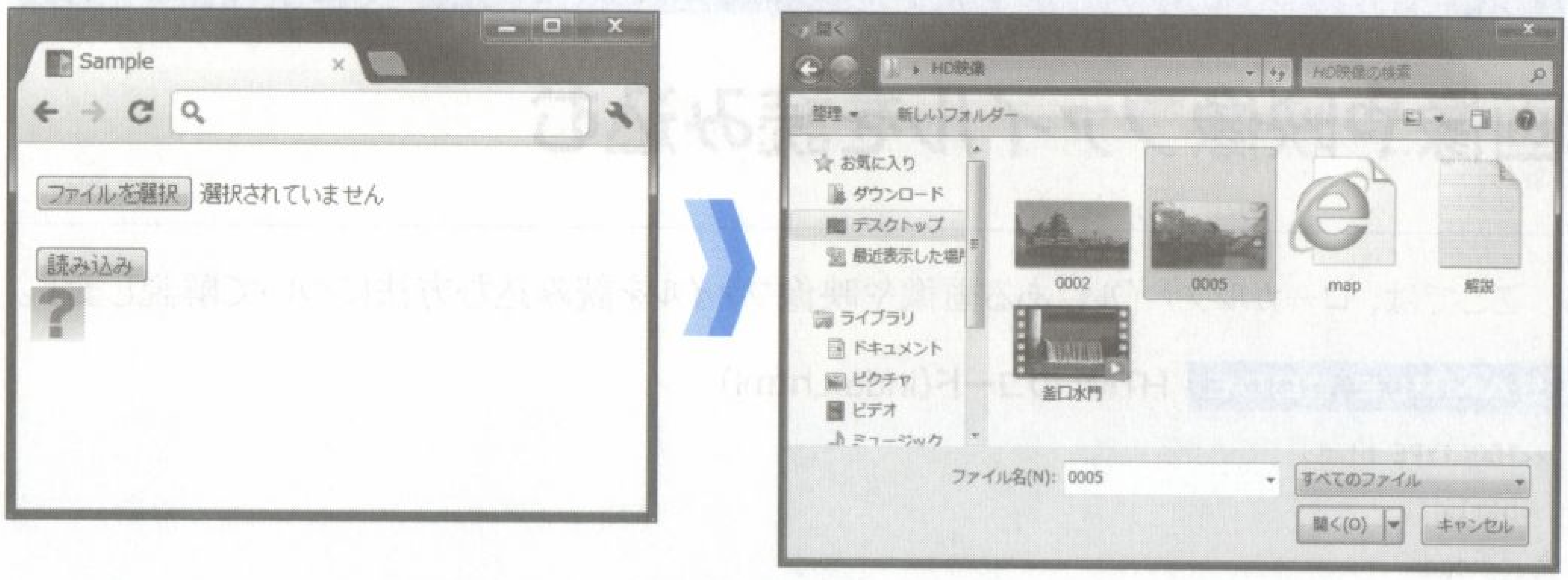
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("readBtn").addEventListener("click", function(){
    var fileList = document.getElementById("selFile").files;
    // FileReaderオブジェクトを作成
    var reader = new FileReader();
    // ファイルが読み込まれた場合の処理を設定
    reader.onload = function(evt){
      var data = evt.target.result;
      // img要素のsrcに代入し表示
      document.getElementsByTagName("img")[0].src = data;
    }
    // ファイルの読み込みを開始
    reader.readAsDataURL(fileList[0]);
  }, false);
}, false);
```

HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示し、ファイル選択ボタンをクリックして、画像ファイルを選択します。



読み込みボタンをクリックすると、選択された画像ファイルが読み込まれ、表示されます。



ONEPOINT　画像ファイルや映像ファイルを読み込むには「readAsDataURL()」メソッドを使う

画像ファイルや映像ファイルを読み込むには、「FileReader」オブジェクトの「readAsDataURL()」メソッドを使います。「readAsDataURL()」のパラメータに、選択されたファイルオブジェクトを指定します。

画像ファイルや映像ファイルの読み込みが完了すると「load」イベントが発生するので、「FileReader」オブジェクトの「onload」にイベントハンドラを設定し、読み込み完了時の処理を行います。

イベントハンドラにはイベントオブジェクトが渡され、「target」オブジェクトの「result」プロパティに読み込まれた画像ファイルや映像ファイルが「data:〜」形式で格納されています。dataURL形式になっているので、表示する場合は画像や映像の「src」属性（プロパティ）に代入します。

関連項目 ▶▶▶

- テキストファイルを読み込む …… p.683
- バイナリファイルを読み込む …… p.685
- ファイルの読み込み状況を表示する …… p.689
- ドラッグ要素からドロップ要素にデータを渡す …… p.866

SECTION-294

ファイルの読み込み状況を表示する

ここでは、ファイルの読み込み状況を表示する方法について解説します。

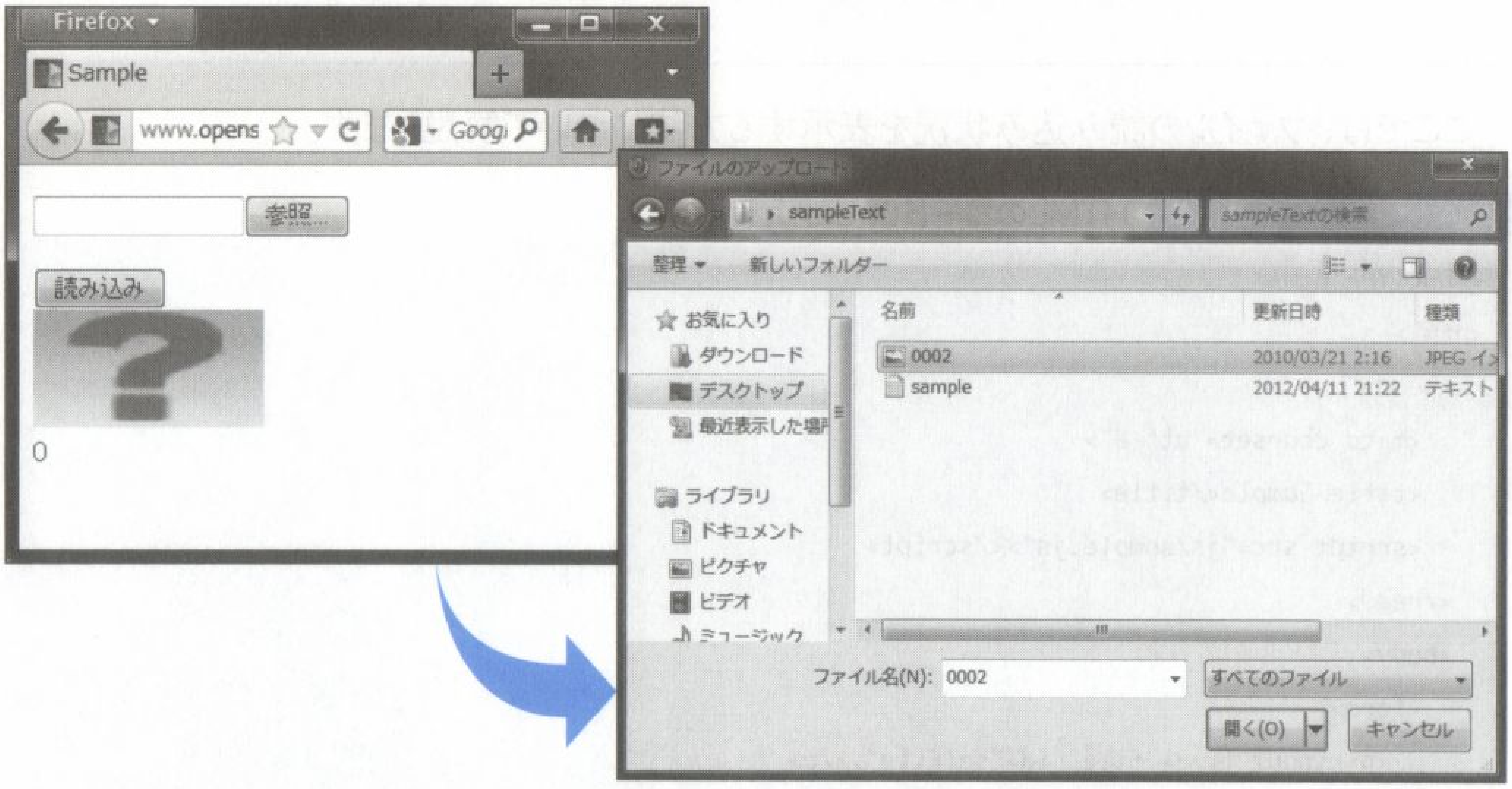
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p><input type="file" id="selFile"></p>
      <input type="button" value="読み込み" id="readBtn">
    </form>
    <div><img src="images/dummy.png" width="128" height="64"></div>
    <output>0</output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタンにイベントを設定
  document.getElementById("readBtn").addEventListener("click", function(){
    var fileList = document.getElementById("selFile").files;
    // FileReaderオブジェクトを作成
    var reader = new FileReader();
    // ファイルが読み込まれた場合の処理を設定
    reader.onload = function(evt){
      var data = evt.target.result;
      // img要素のsrcに代入し表示
      document.getElementsByTagName("img")[0].src = data;
    }
    // 読み込み状況を表示
    reader.onprogress = function(evt){
      // img要素のsrcに代入し表示
      ele.innerHTML = evt.loaded + "バイト読み込みました";
    }
    // ファイルの読み込みを開始
    reader.readAsDataURL(fileList[0]);
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「ファイル選択」ボタンをクリックして、画像ファイルを選択します。



「読み込み」ボタンをクリックすると、読み込んだバイト数と画像が表示されます。



ONEPOINT ファイルの読み込み状況を表示するには「progress」イベントを使う

ファイルの読み込み状況を表示するには、「FileReader」オブジェクトの「progress」イベントを利用します。イベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトの「loaded」プロパティに、読み込まれたファイルのバイト数が入ります。

進行状況を割合で示したい場合には、読み込まれたバイト数をファイルの総バイト数を示すファイルオブジェクトの「size」プロパティで除算します。また、「progress」要素を使って進捗状況を示す方法もあります。

関連項目 ▶▶▶

- テキストファイルを読み込む …… p.683
- バイナリファイルを読み込む …… p.685
- 画像や映像ファイルを読み込む …… p.687

SECTION-295

指定したディスク容量が使用可能か調べる

ここでは、指定したディスク容量が使用可能か調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Quotaチェック</h3>
    <output>0</output>
  </body>
</html>
```

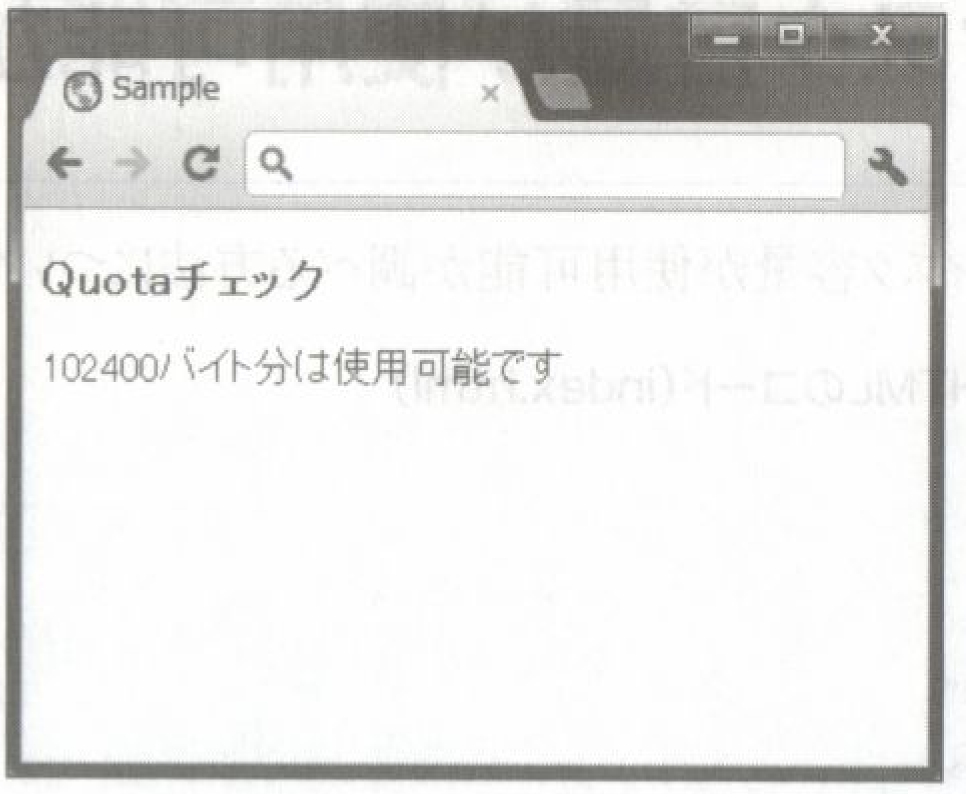
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 指定した容量を確保できるか調べる
  window.storageInfo.requestQuota(
    // 種類を指定(永続的か一時的か)
    TEMPORARY,
    // 確保するサイズ(バイト数)/100KB確保できるか?
    1024*100,
    // 成功時の処理を行うコールバック関数
    function(bytes){
      ele.innerHTML = bytes+"バイト分は使用可能です";
    },
    // 失敗時の処理を行うコールバック関数
    function(err){
      ele.innerHTML = "エラー!!<br>"+err.code;
    }
  );
}, false);
// ベンダープレフィックスも考慮
window.storageInfo = window.storageInfo || window.webkitStorageInfo;
```

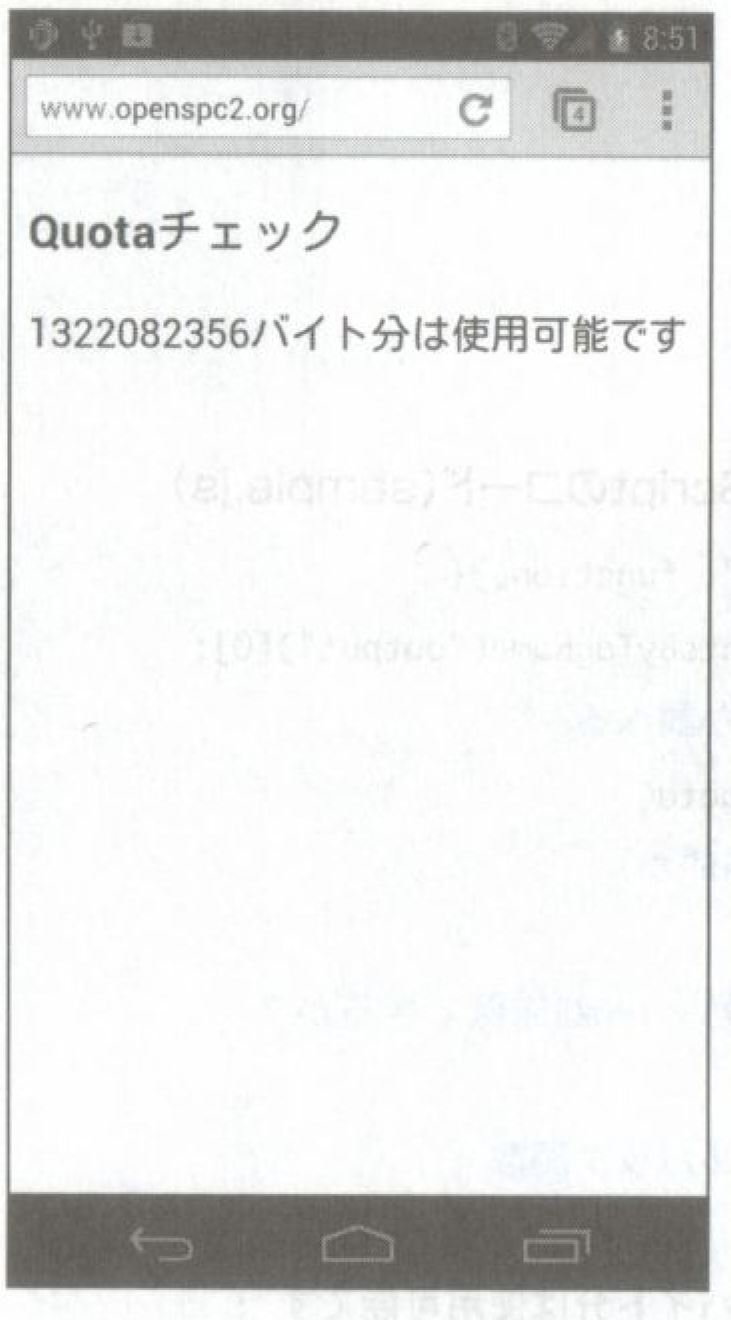
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、使用可能なバイト数が表示されます。



index.htmlをChrome for Androidで表示すると、使用可能なバイト数が表示されます。



ONEPOINT **使用可能な割り当て容量を調べるには「window.storageInfo.requestQuota()」メソッドを使う**

File APIでは、使用可能なディスク容量を確認することができます。調べるには、「window.storageInfo.requestQuota()」メソッドを使います。最初のパラメータには、次の表に示す種類を指定します。なお、文字列ではなく、プロパティとして割り当てられています。

プロパティ(種類)	説明
TEMPORARY	一時的
PERSISTENT	永続的

「TEMPORARY」は、一時的な保存で、全体的な保存領域がなくなった場合などには削除されることがあります。ただし、電源を落としても保存したデータが消えるわけでは

ありません。

2番目のパラメータには、バイト数を指定します。ここで指定したバイト数が確保できた場合に、3番目に指定したパラメータ(コールバック関数)が呼び出されます。失敗した場合は、4番目に指定したパラメータ(コールバック関数)が呼び出されます。

成功時に呼び出されるコールバック関数には、確保したバイト数が渡されます。Google Chromeでは指定したサイズよりも確保できる容量が少ない場合、使用可能な最大バイト数が返されます。

失敗時に呼び出されるコールバック関数には、次の表に示すエラーオブジェクト/コード(「code」プロパティに格納)が渡されます。

エラーコード	キーワード値	説明
1	NOT_FOUND_ERR	ファイルかディレクトリが見つからない
2	SECURITY_ERR	セキュリティエラー
3	ABORT_ERR	中断された
4	NOT_READABLE_ERR	読み込めない
5	ENCODING_ERR	文字のエンコーディングエラー
6	NO_MODIFICATION_ALLOWED_ERR	修正できない
7	INVALID_STATE_ERR	不正なステート
8	SYNTAX_ERR	文法エラー
9	INVALID_MODIFICATION_ERR	不正な修正
10	QUOTA_EXCEEDED_ERR	容量オーバー
11	TYPE_MISMATCH_ERR	不正なファイル形式
12	PATH_EXISTS_ERR	指定されたパスにファイルまたはディレクトリが存在

関連項目 ▶▶▶

●ディスクの使用状況を調べる …… p.694

SECTION-296

ディスクの使用状況を調べる

ここでは、File APIを使って保存できるディスクの使用状況を調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ディスクの使用状況</h3>
    <div>■TEMPORARY<br><output>0</output>バイト</div>
    <div>■PERSISTENT<br><output>0</output>バイト</div>
  </body>
</html>
```

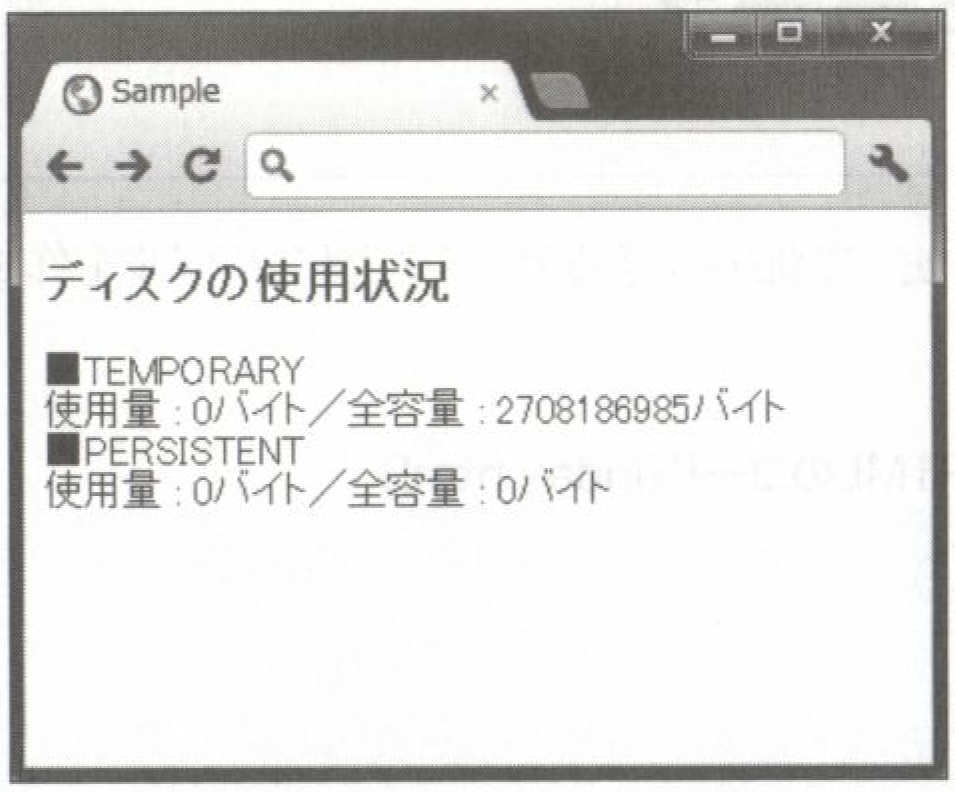
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output");
  checkCapacity(TEMPORARY, 0);  // テンポラリの使用状況を調べる
  checkCapacity(PERSISTENT, 1);  // 永続的に保存される領域の使用状況を調べる
  // 使用状況を調べる
  function checkCapacity(diskType, idx){
    window.storageInfo.queryUsageAndQuota(
      diskType,
      // 成功時の処理を行うコールバック関数
      function(usage, quota){
        ele[idx].innerHTML = "使用量 : "+usage+"バイト/全容量 : "+quota;
      },
      // 失敗時の処理を行うコールバック関数
      function(err){ ele[idx].innerHTML = "エラー:"+err.code; }
    );
  }
}, false);
// ベンダープレフィックスも考慮
window.storageInfo = window.storageInfo || window.webkitStorageInfo;
```

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、全容量と使用量が表示されます。



index.htmlをChrome for Androidで表示すると、全容量と使用量が表示されます。



ONEPOINT ディスクの使用状況を調べるには「window.storageInfo.queryUsageAndQuota()」メソッドを使う

File APIでは、使用可能なディスク容量の使用状況を調べるには、「window.storageInfo.queryUsageAndQuota()」メソッドを使います。最初のパラメータには、次の表に示す保存領域の種類を指定します。これらは、別々の領域に割り当てられています。

プロパティ(種類)	説明
TEMPORARY	一時的
PERSISTENT	永続的

「queryUsageAndQuota()」メソッドの2番目のパラメータには、成功時に呼び出されるコールバック関数を指定します。コールバック関数には、使用しているバイト数と全体のバイト数の2つのパラメータが渡されます。

「queryUsageAndQuota()」メソッドの3番目のパラメータには、エラー発生時に呼び出されるコールバック関数を指定します。パラメータには、エラーコードが渡されます。エラーコードと内容については、693ページを参照してください。

関連項目 ▶▶▶

● 指定したディスク容量が使用可能か調べる ……… p.691

SECTION-297

ファイルを作成する

ここでは、File APIを使って保存できるディスク上にファイルを作成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルを作成</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ファイルの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(1024bytes×100 = 100KB)
    1024*100,
    // 成功時のコールバック関数
    function(fs){
      // sample.txtファイルを生成。すでにある場合は上書き
      fs.root.getFile("sample.txt", { create: true, exclusive: false },
        function(fileEntry){
          ele.innerHTML = fileEntry.name+"ファイルを作成しました<br>";
          ele.innerHTML += "ファイルパス:"+fileEntry.fullPath;
        },
        // 失敗時のコールバック関数
        function(err){ ele.innerHTML = "ファイル生成エラー:"+err.code; }
      );
    },
    // 失敗時のコールバック関数
    function(err){ ele.innerHTML = "エラー:"+err.code; }
  );
}, false);
```

```
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

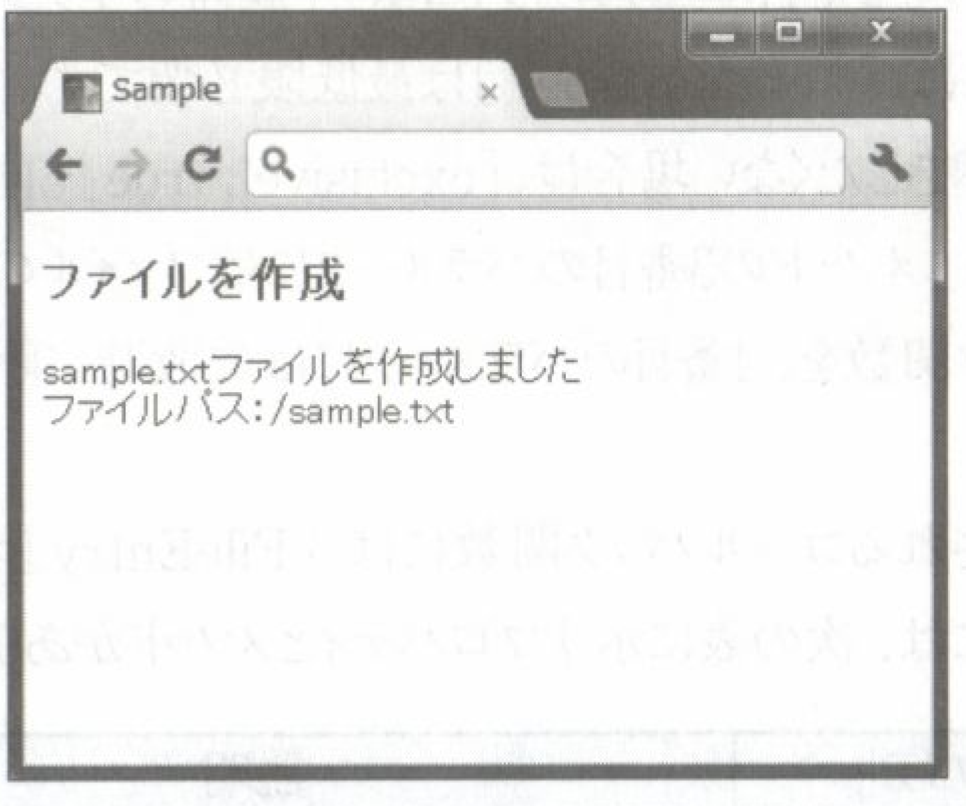
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ファイルが作成され、ファイルパスが表示されます。



index.htmlをChrome for Androidで表示すると、ファイルが作成され、ファイルパスが表示されます。



ONEPOINT

ファイルを作成するには「window.requestFileSystem()」メソッドと「fs.root.getFile()」メソッドを使う

File APIで使用可能なディスク領域にファイルを作成するには、「window.requestFileSystem()」メソッドと「fs.root.getFile()」メソッドを使います。「window.requestFileSystem()」メソッドでは、作成するファイルの保存先の種類やバイト数を指定します。最初のパラメータに保存先の種類(「TEMPORARY」か「PERSISTENT」)、2番目のパラメータにはファイルのバイト数を指定します。

3番目のパラメータには成功時のコールバック関数、4番目のパラメータには失敗時のコールバック関数を指定します。

実際にファイルを作成するには、成功時のコールバック関数内で「fs.root.getFile()」メソッドを利用して行います。「fs.root.getFile()」メソッドの最初のパラメータに、作成するファイル名を指定します。2番目のパラメータには、作成時のオプションを指定します。ファイルを作成するためには、「{ create: true }」は最低限必要です。また、すでに同名のファイルがある場合に上書きしたくない場合は、「exclusive : true」も同時に指定します。

「fs.root.getFile()」メソッドの3番目のパラメータにはファイルの作成が成功した時に呼び出すコールバック関数を、4番目のパラメータには失敗時に呼び出すコールバック関数を指定します。

成功時に呼び出されるコールバック関数には、「FileEntry」オブジェクトが渡されます。このオブジェクトには、次の表に示すプロパティとメソッドがあります。

プロパティ	説明
isFile	ファイルかどうか。「true」ならファイル
isDirectory	ディレクトリかどうか。「true」ならディレクトリ
name	ファイル名
fullPath	ファイルのフルパス
filesystem	「FileSystem」オブジェクト
file	「FileEntry」オブジェクト

メソッド	説明
getMetadata()	メタデータを取得
moveTo()	ファイルを移動
copyTo()	ファイルをコピー
toURL()	「filesystem:」形式(URL)に変換
remove()	ファイルを削除
getParent()	親ディレクトリを取得
createWriter()	ファイルへの書き込み

関連項目 ▶▶▶

- ファイルにテキストを書き込む ……… p.699
- 保存されているファイルの内容を読み出す ……… p.702
- 保存されているファイルを削除する ……… p.705

SECTION-298

ファイルにテキストを書き込む

ここでは、File APIを使って保存できるファイルにテキストを書き込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルにテキストを書き込み</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ファイルの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(1024bytes×100 = 100KB)
    1024*100,
    // 成功時のコールバック関数内でファイル生成を行う
    function(fs){
      // sample.txtファイルを生成。すでにある場合は上書き
      fs.root.getFile("sample.txt", { create: true, exclusive: false },
        function(fileEntry){
          // ファイル書き込み
          fileEntry.createWriter(
            function(fileWriter){
              // 書き込み完了時の処理
              fileWriter.onwrite = function(){
                ele.innerHTML = "書き込み完了";
              }
              // 書き込みできなかった時の処理
              fileWriter.onerror = function(err){
                ele.innerHTML = "書き込みエラー:"+err.code;
              }
```

```
            // Blobを作成して書き込む
            var blobData = new BlobBuilder();
            // 書き込む内容(テキスト)
            blobData.append("File API sample");
            // 標準テキストで書き込む
            fileWriter.write(blobData.getBlob("text/plain"));
          }, fsErr  // 失敗時のコールバック関数
        );
      }, fsErr  // 失敗時のコールバック関数
    );
  }, fsErr  // 失敗時のコールバック関数
  );
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
BlobBuilder = WebKitBlobBuilder;
```

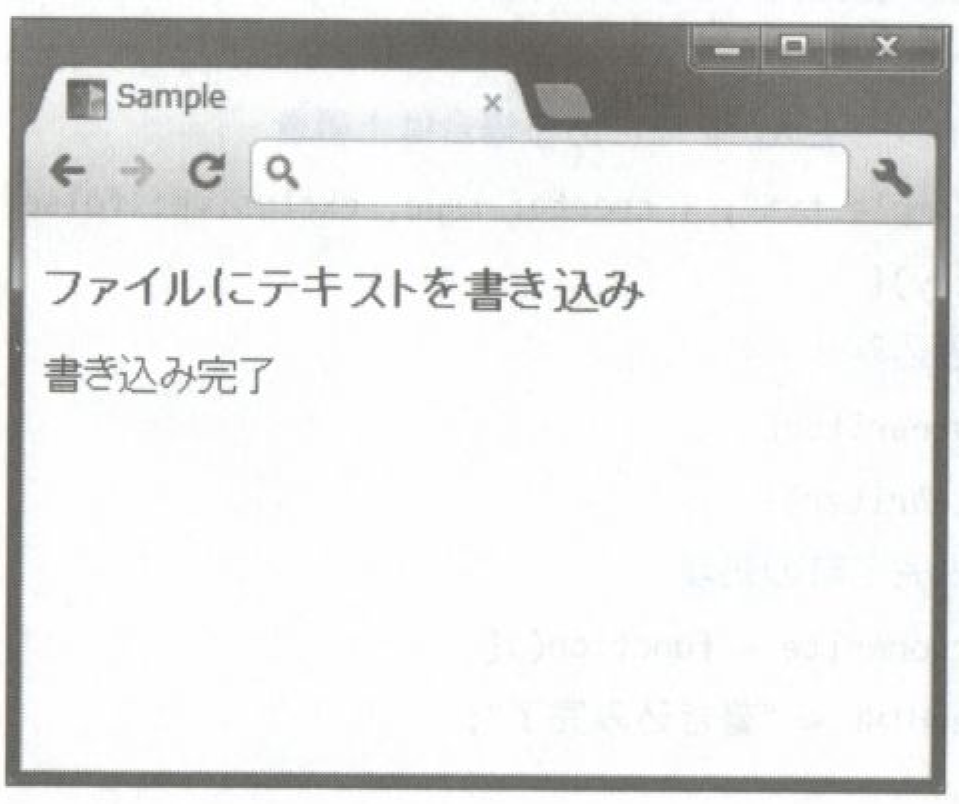
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

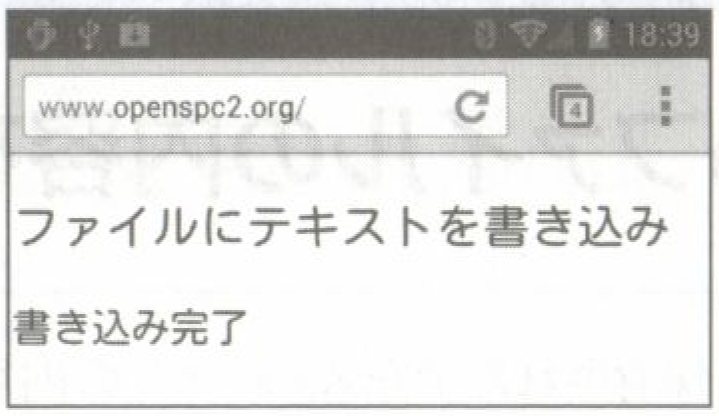
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、正しく書き込まれたことを示すメッセージが表示されます。



index.htmlをChrome for Androidで表示すると、正しく書き込まれたことを示すメッセージが表示されます。



ONEPOINT

ファイルにテキストを書き込むには「createWriter()」メソッドと「BlobBuilder」オブジェクトを使う

ファイルにテキストを書き込むには、「fileEntry」オブジェクトの「createWriter()」メソッドと「BlobBuilder」オブジェクトを使います。「createWriter()」メソッドの最初のパラメータには成功時のコールバック関数を、2番目のパラメータには失敗時のコールバック関数を指定します。

ファイルにテキストを書き込むには、成功時のコールバック関数内で行います。コールバック関数には、「fileWriter」オブジェクトが渡されます。テキストを書き込むには、「Blob」(Binary Large OBject)を使います。書き込むテキストは、「Blob」オブジェクトの「append()」メソッドを利用します。「append()」メソッドのパラメータには、書き込むテキストを指定します。

用意された「Blob」データを、「fileWriter」オブジェクトの「write()」メソッドを使って書き込みます。このとき、「Blob」オブジェクトのデータは「getBlob()」メソッドを使って読み出し、「write()」メソッドのパラメータとして指定します。「getBlob()」メソッドのパラメータには、標準テキストデータである「text/plain」の文字列を指定します。

テキストの書き込みに成功すると、「fileWriter」オブジェクトの「write」イベントが発生します。「fileWriter」オブジェクトの「onwrite」プロパティに、書き込み成功時に呼び出すイベントハンドラを設定することができます。また、書き込めなかった場合には、「error」イベントが発生します。「fileWriter」オブジェクトの「onerror」プロパティに、書き込みエラー発生時に呼び出すイベントハンドラを設定することができます。

COLUMN

「getBlob()」メソッドの動作の違い

「getBlob()」メソッドは、ブラウザによって動作が異なります。Google Chrome(WebKit)では、「getBlob()」メソッドを実行した際に、「append()」メソッドで追加したデータが削除されません。IE10やFirefoxでは追加したデータは、「getBlob()」メソッドによって削除されます。

なお、次のブログ記事を参考にしてください。

URL http://blogs.msdn.com/b/ie_ja/archive/2012/02/01/blobbuilder.aspx

関連項目 ▶▶▶

● ファイルを作成する …… p.696
● 保存されているファイルの内容を読み出す …… p.702

SECTION-299

保存されているファイルの内容を読み出す

ここでは、File APIを使って保存されたテキストファイルの内容を読み出す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>テキストファイルを読み出し</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ファイルの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(読み出しの場合は意味がないので0)
    0,
    // 成功時のコールバック関数内でファイル生成を行う
    function(fs){
      // sample.txtファイルを読み出して表示
      fs.root.getFile("sample.txt", {},
        function(fileEntry){
          fileEntry.file(function(file){
            // FileReaderでファイル読み出し
            var reader = new FileReader();
            // 読み込み完了時の処理(文字をテキストエリアに出力)
            reader.onload = function(){
              ele.innerText = this.result;
            }
            // テキストファイルとして読み出し
            reader.readAsText(file);
          });
        }, fsErr
    );
```

```
    }, fsErr
  );
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

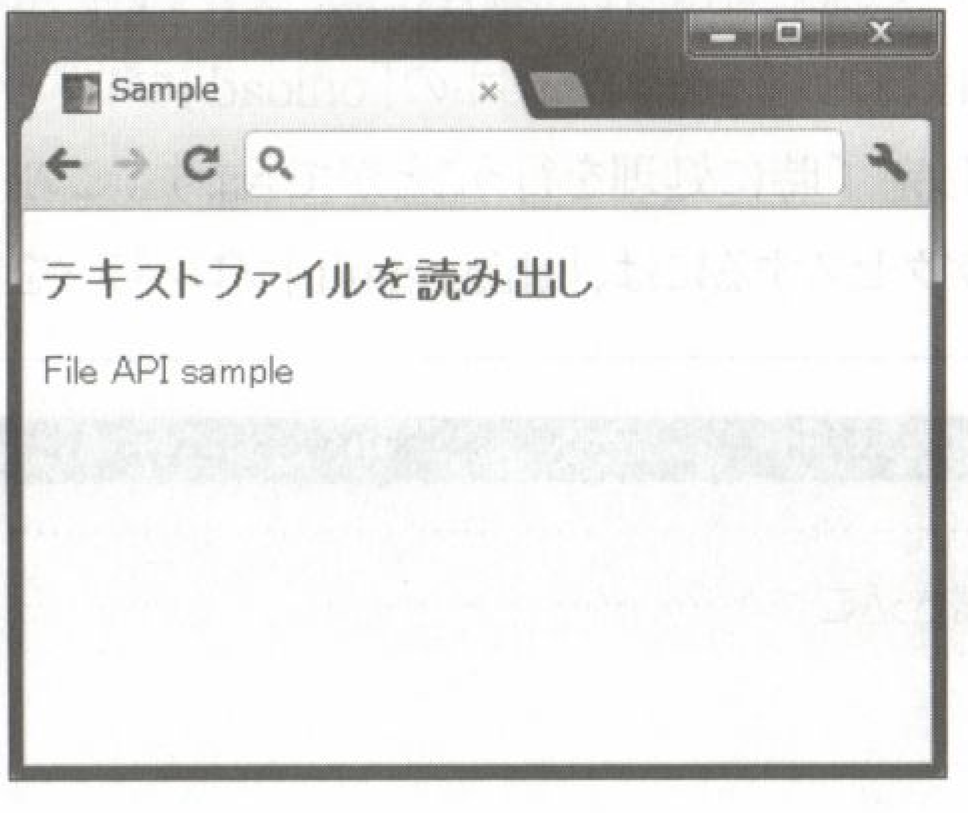
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、sample.txtファイル(内容は「File API sample」の文字)を読み出した結果が表示されます。



index.htmlをChrome for Androidで表示すると、sample.txtファイル(内容は「File API sample」の文字)を読み出した結果が表示されます。



ONEPOINT 保存されているファイルの内容を読み出すには「fs.root.getFile()」メソッドと「FileReader」オブジェクトを使う

保存されているファイルの内容を読み出すには、「fs.root.getFile()」メソッドと「FileReader」オブジェクトを利用します。「fs.root.getFile()」メソッドの最初のパラメータに読み出したいファイルパス(ファイル名)を指定します。ファイルパスは、UNIXのファイルシステムやURLと同様に、「/」(半角スラッシュ)で区切って指定します。2番目のパラメータはオプションですが、不要なので、「{}」として何もオプションを指定しないようにします。

3番目のパラメータは、読み出すファイルが存在した場合に呼び出すコールバック関数になります。4番目のパラメータは、エラー発生時に呼び出すコールバック関数になります。

ファイルの読み出しは、成功時に呼び出すコールバック関数内で行います。コールバック関数には、「fileEntry」オブジェクトが渡されます。ファイルを読み出すには、この「fileEntry」オブジェクトの「file()」メソッドを使います。「file()」メソッドのパラメータに、ファイル内容を読み出す関数を指定します。関数にはファイルオブジェクトが渡されます。ファイルオブジェクトが渡されれば、「FileReader」オブジェクトの「readAsText()」メソッドを使って、テキストファイルを読み出すことができます(683ページ参照)。

テキストファイルがすべて読み出された場合には、ファイルオブジェクトの「load」イベントが発生します。これは、ファイルオブジェクトの「onload」プロパティにイベントハンドラを設定すれば、読み出し完了時に処理を行うことができます。このイベントハンドラ内で読み出されたテキストにアクセスするには、「this.result」プロパティを参照します。

関連項目 ▶▶▶

- ファイルを作成する …… p.696
- ファイルにテキストを書き込む …… p.699

SECTION-300

保存されているファイルを削除する

ここでは、File APIを使って保存されたファイルを削除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルを削除</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(削除の場合は意味がないので0)
    0,
    // 成功時のコールバック関数内でファイル生成を行う
    function(fs){
      // sample.txtファイルを削除
      fs.root.getFile("sample.txt", {},
        function(fileEntry){
          fileEntry.remove(function(){
            ele.innerHTML = fileEntry.fullPath+"を削除しました";
          }, fsErr);
        }, fsErr
      );
    }, fsErr
  );
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

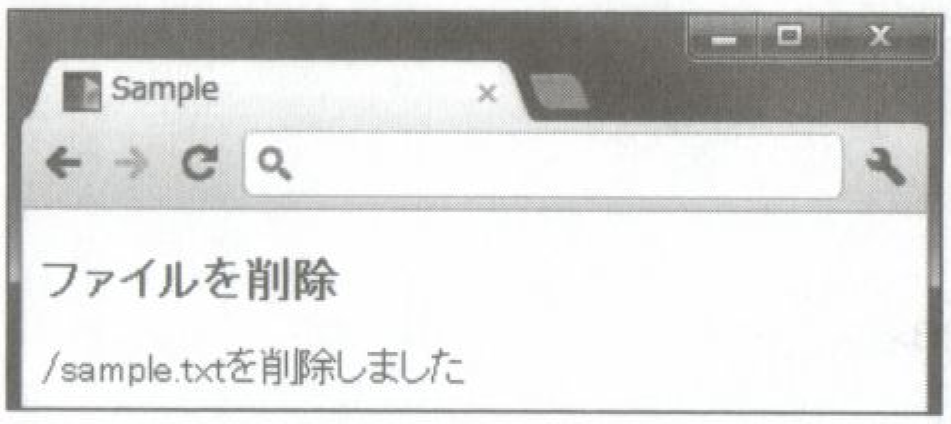
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。また、ルートディレクトリにsample.txtがあることとします。

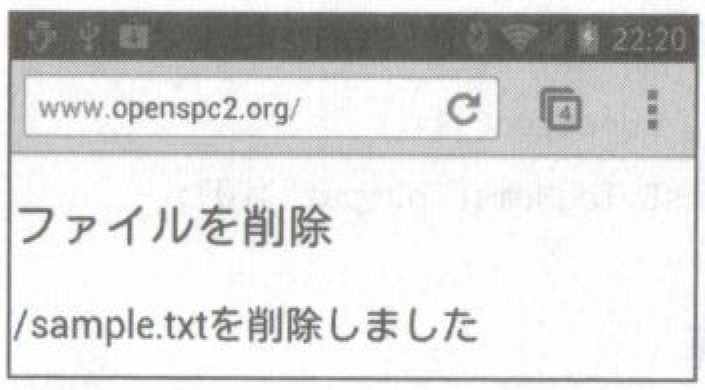
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、sample.txtファイルを削除した結果が表示されます。



index.htmlをChrome for Androidで表示すると、sample.txtファイルを削除した結果が表示されます。



ONEPOINT ファイルを削除するには「fs.root.getFile()」メソッドと「remove()」メソッドを使う

ファイルを削除するには、「fs.root.getFile()」メソッドと「fileEntry」オブジェクトの「remove()」メソッドを利用します。「fs.root.getFile()」メソッドの最初のパラメータに、削除したいファイルパス(ファイル名)を指定します。ファイルパスは、UNIXのファイルシステムやURLと同様に、「/」(半角スラッシュ)で区切って指定します。2番目のパラメータはオプションですが、不要なので、「{}」として何もオプションを指定しないようにします。

3番目のパラメータは、削除するファイルが存在した場合に呼び出すコールバック関数になります。4番目のパラメータは、エラー発生時に呼び出すコールバック関数になります。

ファイルの削除は、成功時に呼び出すコールバック関数内で行います。コールバック関数には「fileEntry」オブジェクトが渡されます。ファイルを削除するには、この「fileEntry」オブジェクトの「remove()」メソッドを使います。「remove()」メソッドのパラメータにファイル削除後に実行する関数を指定します。なお、削除するファイルがない場合は、エラーになります。

関連項目 ▶▶▶

● ファイルを作成する ……………………………… p.696

SECTION-301

ディレクトリを作成する

ここでは、File APIでディレクトリを作成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ディレクトリを作成</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ディレクトリの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(1024bytes×100 = 100KB)
    1024*100,
    // 成功時のコールバック関数
    function(fs){
      // Documentディレクトリを生成。すでにある場合は上書き
      fs.root.getDirectory("Document", { create: true },
        function(dirEntry){
          ele.innerHTML = dirEntry.name+"ディレクトリを作成しました<br>";
          ele.innerHTML += "ディレクトリパス:"+dirEntry.fullPath;
        },
        // 失敗時のコールバック関数
        function(err){ ele.innerHTML = "ファイル生成エラー:"+err.code; }
      );
    },
    // 失敗時のコールバック関数
    function(err){ ele.innerHTML = "エラー:"+err.code; }
  );
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

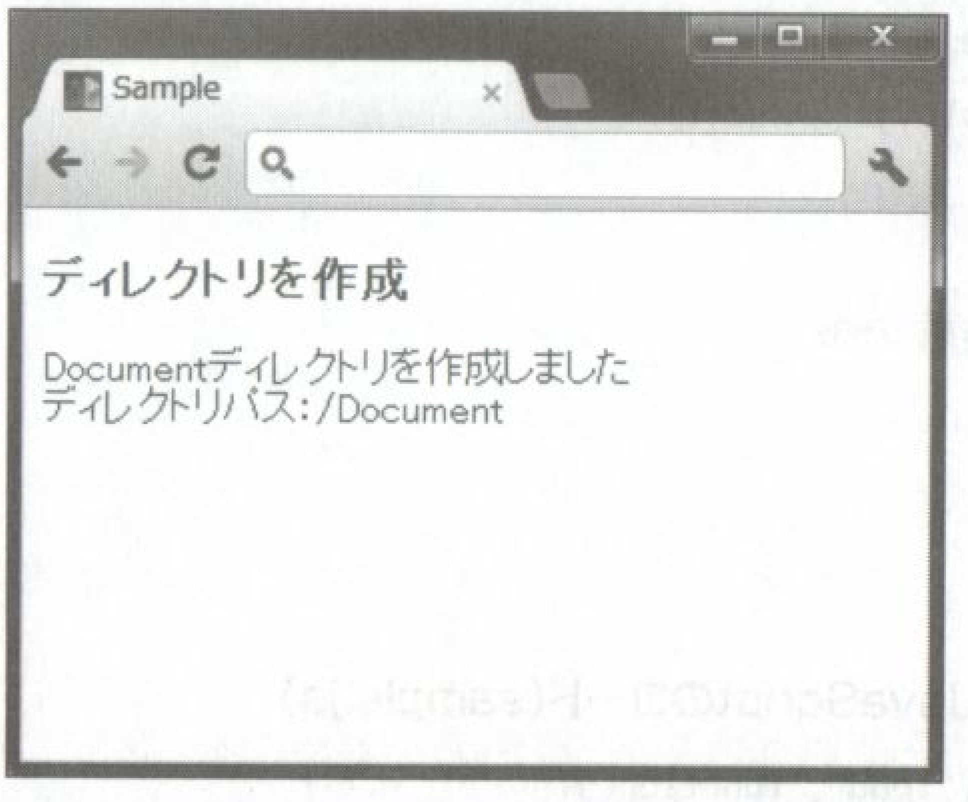
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

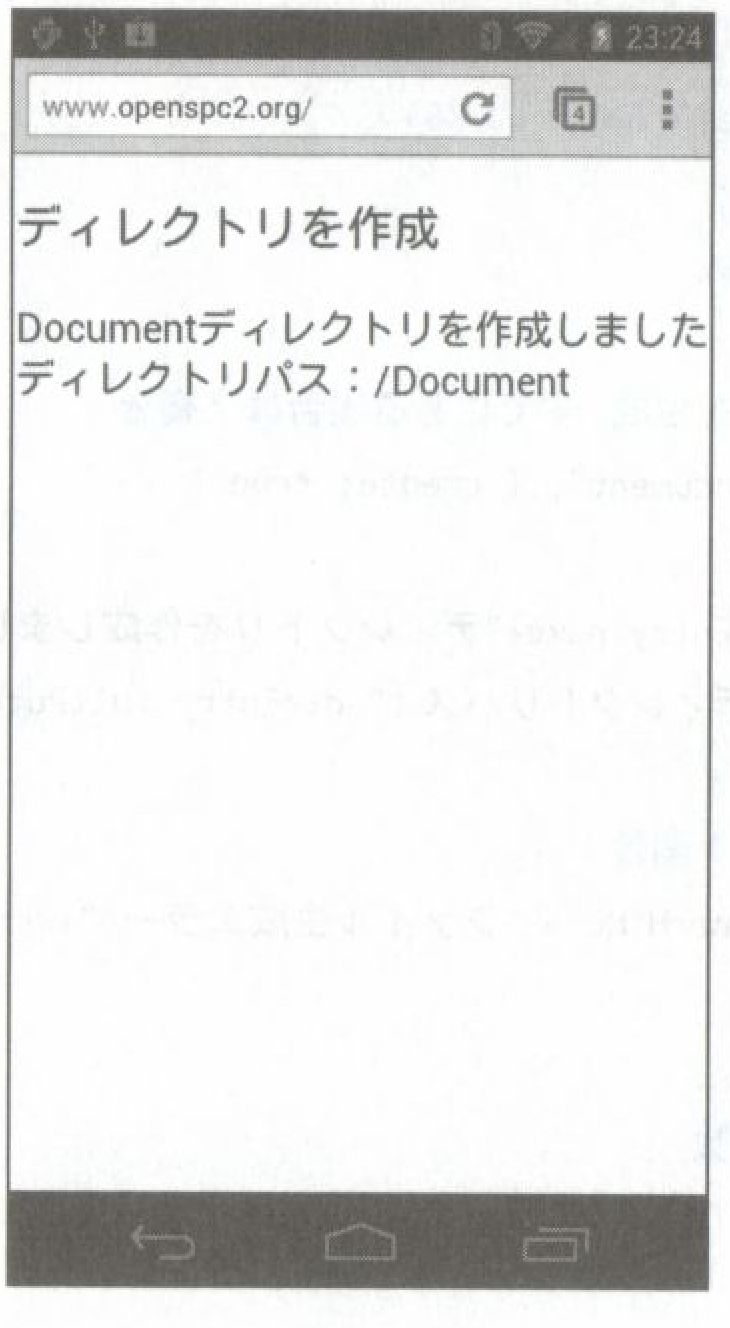
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ディレクトリが作成された場合、メッセージが表示されます。



index.htmlをChrome for Androidで表示すると、ディレクトリが作成された場合、メッセージが表示されます。



ONEPOINT

ディレクトリを作成するには「window.requestFileSystem()」メソッドと「fs.root.getDirectory()」メソッドを使う

File APIで使用可能なディスク領域にディレクトリを作成するには、「window.requestFileSystem()」メソッドと「fs.root.getDirectory()」メソッドを使います。「window.requestFileSystem()」メソッドでは、作成するディレクトリの保存領域の種類やバイト数を指定します。最初のパラメータに保存先の種類(「TEMPORARY」か「PERSISTENT」)、2番目のパラメータにはディレクトリのバイト数を指定します。

3番目のパラメータには成功時のコールバック関数、4番目のパラメータには失敗時のコールバック関数を指定します。

実際のディレクトリの作成は、成功時のコールバック関数内で「fs.root.getDirectory()」メソッドを利用して行います。「fs.root.getDirectory()」メソッドの最初のパラメータに、作成するディレクトリ名を指定します。2番目のパラメータには、作成時のオプションである「create: true」を指定します。

「fs.root.getDirectory()」メソッドの3番目のパラメータにはディレクトリの作成が成功したときに呼び出すコールバック関数を、4番目のパラメータには失敗したときに呼び出すコールバック関数を指定します。

成功時に呼び出されるコールバック関数には、「dirEntry」オブジェクトが渡されます。このオブジェクトには、次の表に示すプロパティとメソッドがあります。

プロパティ	説明
isFile	ファイルかどうか。「true」ならファイル
isDirectory	ディレクトリかどうか。「true」ならディレクトリ
name	ディレクトリ名
fullPath	ディレクトリのフルパス
filesystem	「FilsSystem」オブジェクト

メソッド	説明
getMetadata()	メタデータを取得
moveTo()	ディレクトリを移動
copyTo()	ディレクトリをコピー
toURL()	「filesystem:」形式(URL)に変換
remove()	ディレクトリを削除
removeRecursively()	サブディレクトリも含めてまるごと削除
getParent()	親ディレクトリを取得
createReader()	「DirectoryReader」オブジェクトの生成
getFile()	ファイル一覧を取得
getDirectory()	ディレクトリ一覧を取得

関連項目 ▶▶▶

● サブディレクトリを作成する……p.710
● ディレクトリを削除する……p.713

SECTION-302

サブディレクトリを作成する

ここでは、File APIでサブディレクトリを作成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>サブディレクトリを作成</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ディレクトリの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(1024bytes×100 = 100KB)
    1024*100,
    // 成功時のコールバック関数
    function(fs){
      // 各種ディレクトリを生成
      makeSubDirectory(fs.root, "temp");
      makeSubDirectory(fs.root, "temp/photo");
      makeSubDirectory(fs.root, "temp/music");
      makeSubDirectory(fs.root, "temp/photo/2012");
    }, fsErr
  );
  // サブディレクトリを作成。常に上書きで作成
  function makeSubDirectory(baseDir, dirName){
    baseDir.getDirectory(dirName, { create: true },
      function(dirEntry){
        ele.innerHTML += dirEntry.name+"ディレクトリを作成しました<br>";
        ele.innerHTML += "ディレクトリパス :"+dirEntry.fullPath+"<br>";
      }, fsErr
    );
  }
```

```
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、サブディレクトリが作成された場合、メッセージが表示されます。



index.htmlをChrome for Androidで表示すると、サブディレクトリが作成された場合、メッセージが表示されます。



ONEPOINT **サブディレクトリを作成するには「getDirectory()」メソッドを使う**

File APIで使用可能なディスク領域にサブディレクトリを作成するには、「getDirectory()」メソッドを使います。サブディレクトリを作成するには、「getDirectory()」メソッドの最初のパラメータに、「/」(スラッシュ)で区切って作成したいディレクトリ名を記述します。ただし、親ディレクトリが存在していない場合は作成されず、エラーになります。この点以外は、通常のディレクトリ作成と同じです(707ページ参照)。

関連項目 ▶▶▶

- ディレクトリを作成する ……………………………… p.707
- ディレクトリを削除する ……………………………… p.713

SECTION-303

ディレクトリを削除する

ここでは、File APIを使って作成されたディレクトリを削除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ディレクトリを削除</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(削除の場合は意味がないので0)
    0,
    // 成功時のコールバック関数内で削除処理を行う
    function(fs){
      // tempディレクトリを削除
      fs.root.getDirectory("temp", {},
        function(dirEntry){
          dirEntry.removeRecursively(function(){
            ele.innerHTML = dirEntry.fullPath+"を削除しました";
          }, fsErr);
        }, fsErr
      );
    }, fsErr
  );
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

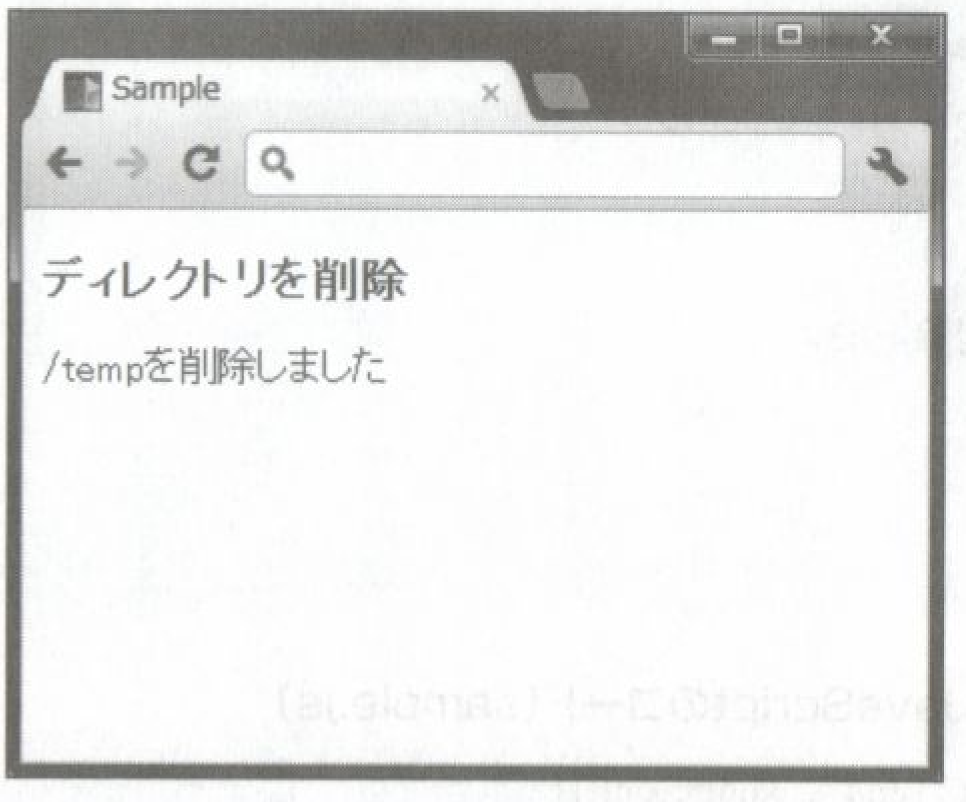
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。また、「temp」ディレクトリは作成済みのこととします。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、「temp」ディレクトリが削除され、メッセージが表示されます。



index.htmlをChrome for Androidで表示すると、「temp」ディレクトリが削除され、メッセージが表示されます。



ディレクトリがすでに削除されている場合に、再度、削除しようとすると、エラーになります。



ONEPOINT ディレクトリを削除するには「fs.root.getDirectory()」メソッドと「remove()」メソッドや「removeRecursively()」メソッドを使う

ディレクトリを削除するには、「fs.root.getDirectory()」メソッドと「dirEntry」オブジェクトの「remove()」メソッド、または、「removeRecursively()」メソッドを利用します。「fs.root.getDirectory()」メソッドの最初のパラメータに、削除したいディレクトリ名を指定します。

3番目のパラメータは、削除するディレクトリが存在した場合に呼び出すコールバック関数になります。4番目のパラメータは、エラー発生時に呼び出すコールバック関数になります。

ディレクトリを削除するには、成功時に呼び出すコールバック関数内で行います。コールバック関数には、「dirEntry」オブジェクトが渡されます。ディレクトリを削除するには、この「dirEntry」オブジェクトの「remove()」メソッド、または、「removeRecursively()」メソッドを使います。「remove()」メソッド、または、「removeRecursively()」メソッドのパラメータに、ディレクトリ削除後に実行する関数を指定します。

「remove()」メソッドは、該当するディレクトリだけを削除します。その際、ディレクトリ内が空である必要があります。「removeRecursively()」メソッドの場合は、ディレクトリ内にあるファイルやサブディレクトリなども含めてまるごと削除します。なお、削除するディレクトリがない場合は、エラーになります。

関連項目 ▶▶▶

- ディレクトリを作成する ……………… p.707
- サブディレクトリを作成する ……………… p.710

SECTION-304

ディレクトリ一覧を取得する

ここでは、File APIを使って作成されたディレクトリ内にあるファイルなどの一覧を取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ディレクトリ一覧を表示</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(一覧取得の場合は意味がないので0)
    0,
    // 成功時のコールバック関数内で処理を行う
    function(fs){
      // ルートディレクトリ一覧を取得
      fs.root.getDirectory("/", {},
        function(dirEntry){
          // ディレクトリ内を読み出すオブジェクトを生成
          var dirReader = dirEntry.createReader();
          // 一覧を取得
          dirReader.readEntries(
            function(list){
              // ファイル/ディレクトリの数だけ繰り返し
              for(var i=0; i<list.length; i++){
                ele.innerHTML += list[i].name;
                if (list[i].isFile){
                  // ファイルの場合
                  ele.innerHTML += "ファイル<br>";
                }else{
```

```
                    // ディレクトリの場合
                    ele.innerHTML += "ディレクトリ<br>";
                  }
                }
              }, fsErr
            );
          }, fsErr
        );
      }, fsErr
    );
    // エラーを一括して処理
    function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

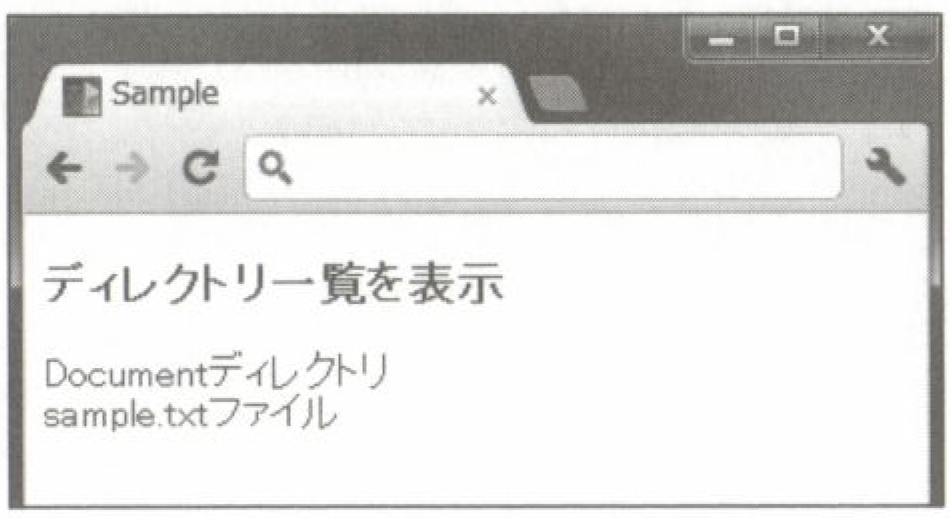
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ルートディレクトリ内にあるファイル・フォルダの一覧が表示されます。



index.htmlをChrome for Androidで表示すると、ルートディレクトリ内にあるファイル・フォルダの一覧が表示されます。



ONEPOINT **ディレクトリ一覧を取得するには「createReader()」メソッドを使う**

ディレクトリ一覧を取得するには、「dirEntry」オブジェクトの「createReader()」メソッドを呼び出します。返されたオブジェクトの「readEntries()」メソッドを使うことで、ディレクトリ内にあるファイル・フォルダの一覧が取得できます。「readEntries()」メソッドのパラメータには、取得に成功したときに呼び出すコールバック関数を指定します。このコールバック関数のパラメータに、ファイル・ディレクトリの一覧が配列として渡されます。配列には、「File」オブジェクトや「Direcotry」オブジェクトが入っているので、必要なプロパティを読み出します。

SECTION-305

ファイルをコピーする

ここでは、File APIを使って作成されたファイルをコピーする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルをコピーする</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(ここでは意味がないので0)
    0,
    // 成功時のコールバック関数内で処理を行う
    function(fs){
      // test.txtをコピーする
      fs.root.getFile("/test.txt", {},
        function(fileEntry){
          fileEntry.getParent(function(targetDir){
            // testCopy時間という新たな名前でコピーを作成
            var time = Date.now();
            var newFileName = "testCopy"+time+".txt";
            fileEntry.copyTo(targetDir, newFileName, function(copyfile){
              ele.innerHTML = copyfile.name+"としてコピーしました。";
            }, fsErr);
          }
        , fsErr);
      }, fsErr
    );
  }, fsErr
);
```

```
    // エラーを一括して処理
    function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

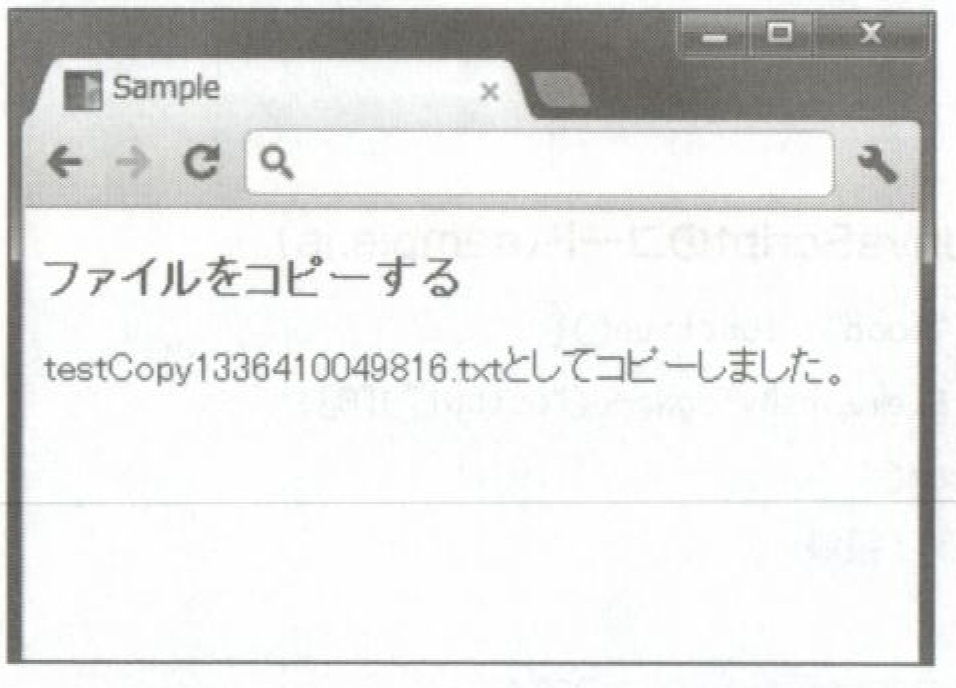
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。また、ルートディレクトリにtest.txtがあることとします。

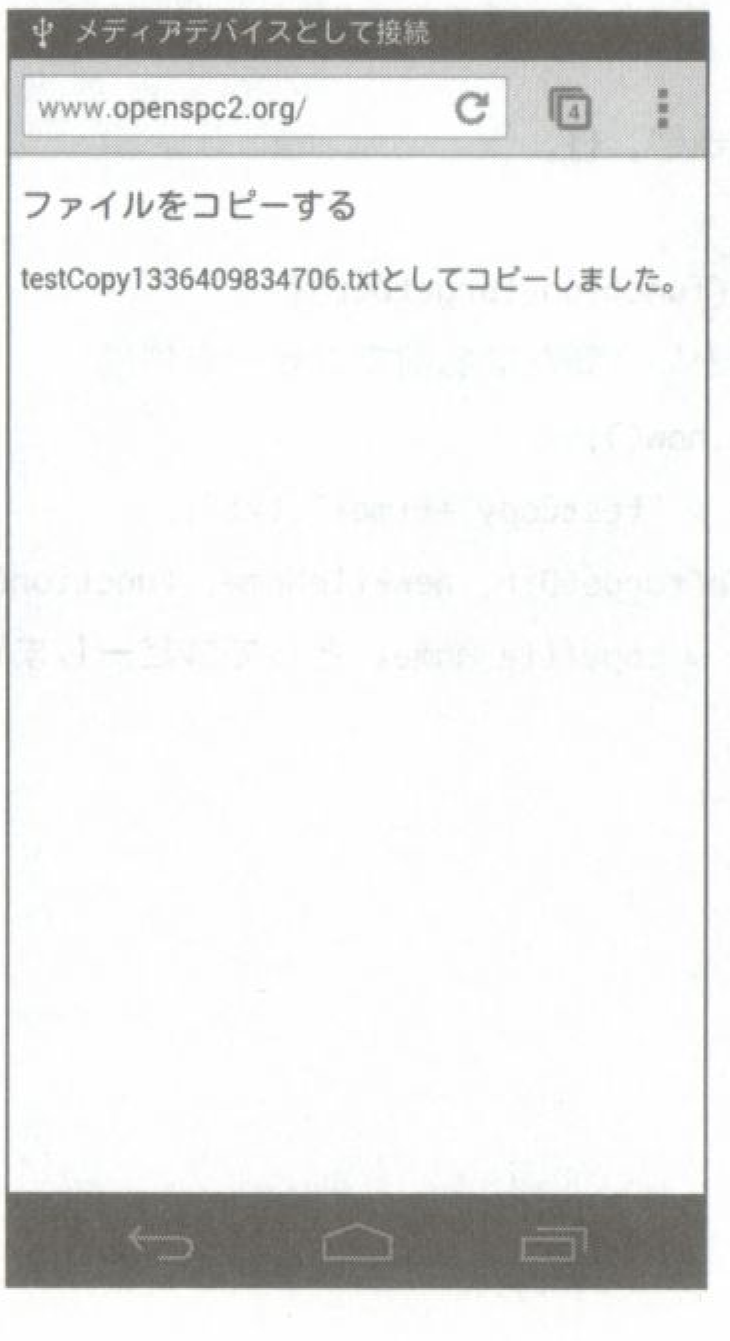
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ルートディレクトリ上にあるtest.txtファイルがコピーされ、コピーしたファイル名が表示されます。



index.htmlをChrome for Androidで表示すると、ルートディレクトリ上にあるtest.txtファイルがコピーされ、コピーしたファイル名が表示されます。



ONEPOINT

ファイルをコピーするには「copyTo()」メソッドと「getParent()」メソッドを使う

ファイルをコピーするには、「fileEntry」オブジェクトの「copyTo()」メソッドと「getParent()」メソッドを呼び出します。「copyTo()」メソッドが実際にコピーを行いますが、その際、親ディレクトリ情報が必要になります。その親ディレクトリ情報を取得するのが、「getParent()」メソッドです。「getParent()」メソッドのパラメータには、成功時のコールバック関数を指定します。このコールバック関数に、取得した親ディレクトリオブジェクトが渡されます。

次に「fileEntry」オブジェクトの「copyTo()」メソッドを使い、ファイルのコピーを行います。最初のパラメータに親ディレクトリを示すオブジェクトを指定します。2番目のパラメータには、コピーした際の新たなファイル名を指定します。異なるディレクトリに同名のファイルをコピーする場合は、2番目のパラメータは省略することができます。

「copyTo()」メソッドの3番目のパラメータには、コピーに成功したときのコールバック関数を指定します。パラメータには、「fileEntry」オブジェクトが渡されます。「copyTo()」メソッドの4番目のパラメータには、エラー発生時のコールバック関数を指定します。

関連項目 ▶▶▶

● ディレクトリをコピーする……………… p.722
● ファイルの移動とファイル名の変更を行う……………… p.725

SECTION-306

ディレクトリをコピーする

ここでは、File APIを使って作成されたディレクトリをコピーする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ディレクトリをコピーする</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(ここでは意味がないので0)
    0,
    // 成功時のコールバック関数内で処理を行う
    function(fs){
      // Documentディレクトリをコピーする
      fs.root.getDirectory("/Document", {},
        function(dirEntry){
          dirEntry.getParent(function(targetDir){
            // Document時間という新たな名前でコピーを作成
            var time = Date.now();
            var newFileName = "Document"+time;
            dirEntry.copyTo(targetDir, newFileName, function(copyfile){
              ele.innerHTML = copyfile.name+"としてコピーしました。";
            }, fsErr);
          }
        , fsErr);
      }, fsErr
    );
  }, fsErr
);
```

```
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

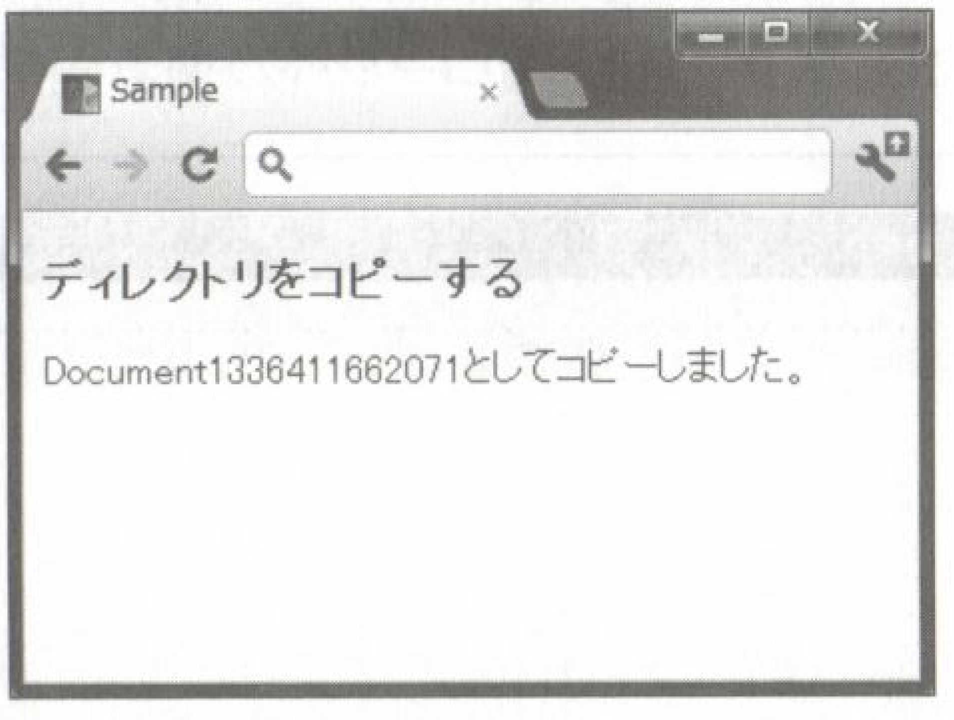
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。また、ルートディレクトリにDocumentディレクトリがあることとします。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ルートディレクトリ上にあるDocumentディレクトリがコピーされ、コピーしたディレクトリ名が表示されます。



index.htmlをChrome for Androidで表示すると、ルートディレクトリ上にあるDocumentディレクトリがコピーされ、コピーしたディレクトリ名が表示されます。



ONEPOINT ディレクトリをコピーするには「copyTo()」メソッドと「getParent()」メソッドや「getDirectory()」メソッドを使う

ディレクトリをコピーするには、「dirEntry」オブジェクトの「copyTo()」メソッドと、「getParent()」メソッド、または、「getDirectory()」メソッドを使います。「copyTo()」メソッドが実際にコピーを行いますが、その際、親ディレクトリ情報が必要になります。その親ディレクトリ情報を取得するのが、「getParent()」メソッドです。「getParent()」メソッドのパラメータには、成功時のコールバック関数を指定します。このコールバック関数に取得した親ディレクトリオブジェクトが渡されます。また、「getDirectory()」メソッドを使って、指定したディレクトリを対象にする方法もあります。

次に「dirEntry」オブジェクトの「copyTo()」メソッドを使い、ファイルのコピーを行います。最初のパラメータに、親ディレクトリを示すオブジェクトを指定します。2番目のパラメータには、コピーした際の新たなファイル名を指定します。異なるディレクトリに同名のファイルをコピーする場合は、2番目のパラメータは省略することができます。

「copyTo()」メソッドの3番目のパラメータには、コピーに成功したときのコールバック関数を指定します。パラメータには、「dirEntry」オブジェクトが渡されます。「copyTo()」メソッドの4番目のパラメータには、エラー発生時のコールバック関数を指定します。

関連項目 ▶▶▶

● ファイルをコピーする ……………………………… p.719

ファイルの移動とファイル名の変更を行う

ここでは、File APIを使って作成されたファイルを移動する方法とファイル名を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルを移動する</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // 処理対象はテンポラリ領域
    TEMPORARY,
    // 確保するサイズ(ここでは意味がないので0)
    0,
    // 成功時のコールバック関数内で処理を行う
    function(fs){
      // test.txtを移動する
      fs.root.getFile("/test.txt", {},
        function(fileEntry){
          // Documentディレクトリに移動
          fs.root.getDirectory("Document", {}, function(targetDir){
              // 同じ名前で移動
              fileEntry.moveTo(targetDir, "test.txt", function(movefile){
                // 移動先のファイルパスを取得
                var fullPath = movefile.fullPath;
                ele.innerHTML = movefile.name+"を"+fullPath+"に移動しました。";
              }, fsErr);

            }
          , fsErr);
        }, fsErr
```

```
      );
    }, fsErr
  );
  // エラーを一括して処理
  function fsErr(err){ alert("エラー:"+err.code); }
}, false);
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

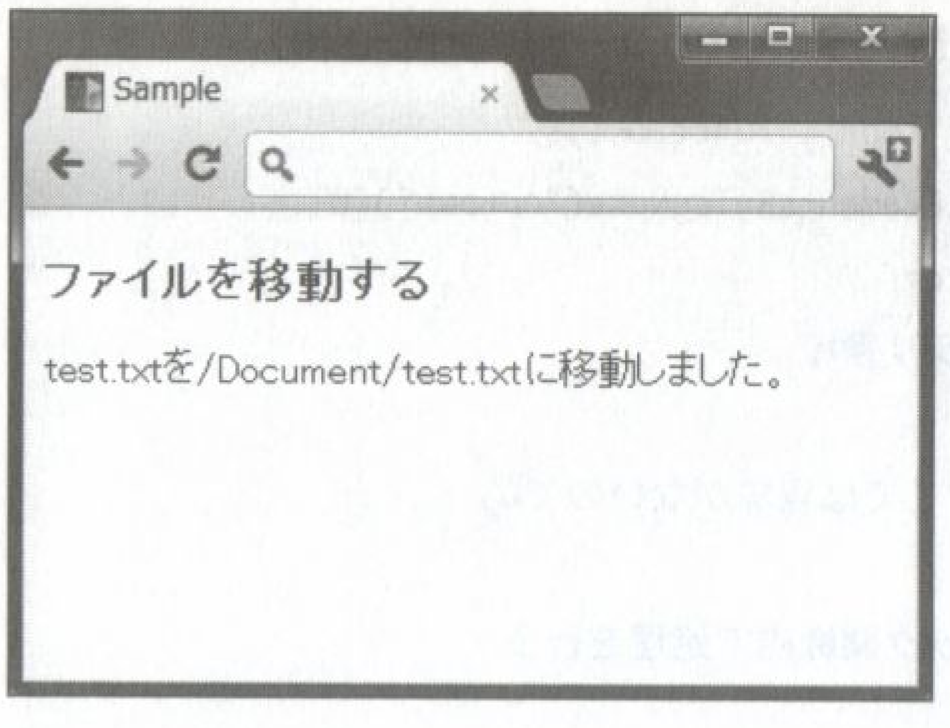
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。また、ルートディレクトリにtest.txtとDocumentディレクトリがあることとします。

HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、ルートディレクトリにあるtest.txtが、同じルートディレクトリ上にあるDocument内に移動され、ファイル名と移動先のパスが表示されます。



index.htmlをChrome for Androidで表示すると、ルートディレクトリにあるtest.txtが、同じルートディレクトリ上にあるDocument内に移動され、ファイル名と移動先のパスが表示されます。



ONEPOINT **ファイルの移動やファイル名の変更には「moveTo()」メソッドを使う**

ファイルを移動するには、「fileEntry」オブジェクトの「moveTo()」メソッドを使います。「moveTo()」メソッドは2つの機能があります。1つはファイルを移動する機能、もう1つがファイル名を変更する機能です。「moveTo()」メソッドでは移動先が同じディレクトリで新しい名前が指定された場合、ファイル名の変更処理を行います。移動先が異なるディレクトリの場合は、ファイルをそのディレクトリに移動させます。

「moveTo()」メソッドの最初のパラメータには、基準となるディレクトリ情報が必要になります。ディレクトリ情報を取得には、「getParent()」メソッド、または、「getDirectory()」メソッドを使います。

「moveTo()」メソッドの2番目のパラメータを省略した場合は、最初のパラメータで指定したディレクトリに同じ名前でファイルが移動します。2番目のパラメータに名前を指定すると、それが新たなファイル名となります。

「moveTo()」メソッドの3番目のパラメータには、移動、または、名前の変更に成功したときのコールバック関数を指定します。パラメータには、「fileEntry」オブジェクトが渡されます。「moveTo()」メソッドの4番目のパラメータには、エラー発生時のコールバック関数を指定します。

関連項目 ▶▶▶

● ファイルをコピーする ……………… p.719

SECTION-308

URL形式でファイルにアクセスする

ここでは、File APIを使って保存できるディスク上にあるファイルをURL形式でアクセスする方法について解説します。

SAMPLE CODE　HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ファイルのURLを表示</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE　JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  window.requestFileSystem(
    // ファイルの作成先をテンポラリ領域にする
    TEMPORARY,
    // 確保するサイズ(1024bytes×100 = 100KB)
    1024*100,
    // 成功時のコールバック関数
    function(fs){
      // sample.txtファイル情報を取得
      fs.root.getFile("sample.txt", { },
        function(fileEntry){
          ele.innerHTML = fileEntry.name+"ファイルのURLは以下の通りです<br>";
          ele.innerHTML += fileEntry.toURL();
        },
        // 失敗時のコールバック関数
        function(err){ ele.innerHTML = "ファイル生成エラー:"+err.code; }
      );
    },
    // 失敗時のコールバック関数
    function(err){ ele.innerHTML = "エラー:"+err.code; }
  );
}, false);
```

```
// ベンダープレフィックスも考慮
window.requestFileSystem = window.requestFileSystem || window.webkitRequestFileSystem;
```

HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で確認してください。

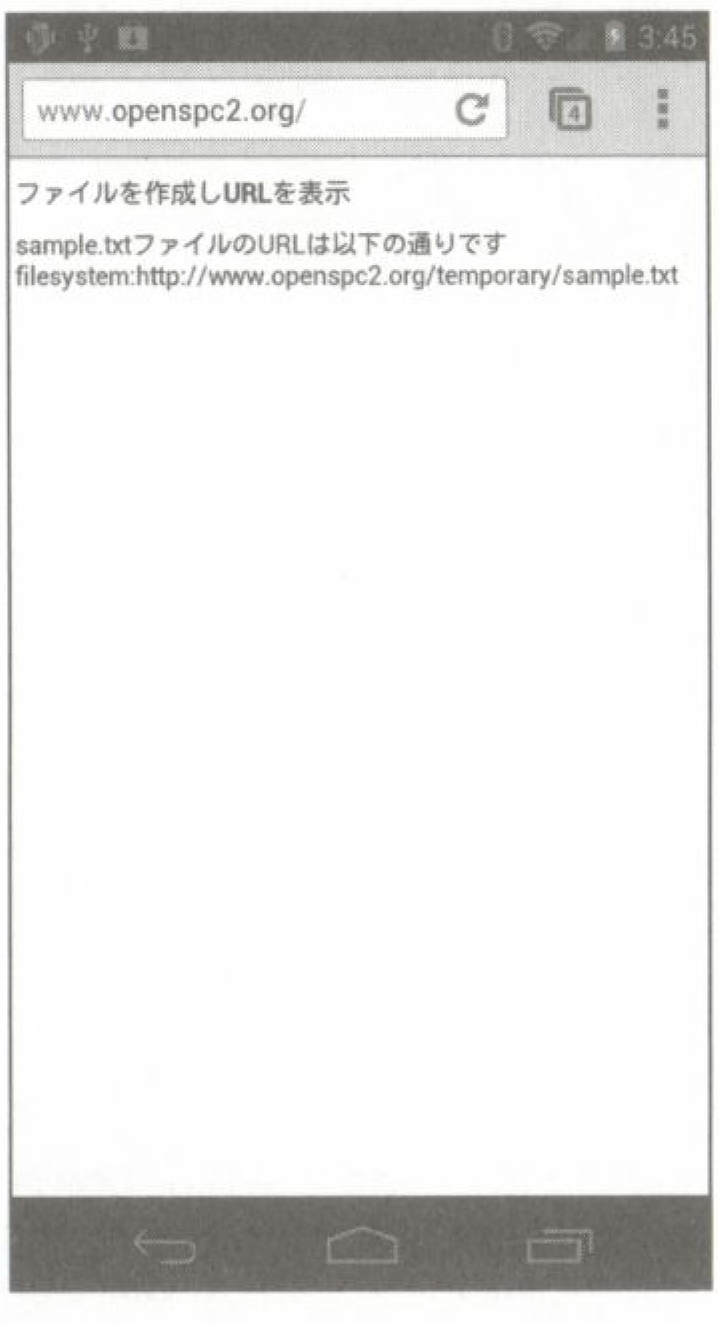
HINT

Androidでは、Chrome for Androidで対応しています。Android 4.1からはデフォルトのブラウザがChromeになっているので、そのまま動作します。

index.htmlをGoogle Chromeで表示すると、指定したファイルのURLが表示されます。



index.htmlをChrome for Androidで表示すると、指定したファイルのURLが表示されます。



ONEPOINT ファイルやディレクトリはURL形式でアクセスすることができる

File APIで使用可能なディスク領域に保存されているファイルやディレクトリはURL形式でアクセスすることができます。これは、次の形式で指定します。

```
filsystem:【オリジン】/【種類】/
```

オリジンは「http://www.openspc2.org:8080/」といったドメインとポート番号の組み合わせです。種類は永続的に保存する領域のファイルの場合は「persistent」、一時的な領域の場合は「temporary」になります。たとえば、一時的に保存されている領域のルートディレクトリにある「sample.txt」であれば、次のようになります。

```
filesystem:http://www.openspc2.org/persistent/sample.txt
```

Google Chromeでは、このURLをアドレス欄に入力すればファイル内容を確認することもできます。また、画像であれば、「img」要素の「src」属性に指定することもできます。この場合、File APIで保存されている領域内にあるデータが読み出され使用されます。

関連項目 ▶▶▶

- Base64エンコード/デコードする ……… p.272
- ページのURLを取得・設定する ……… p.307
- Canvasの画像データをdataURL形式に変換する ……… p.527

CHAPTER 13

HTML5の
その他のAPI

IE 9 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-309

一度だけ位置情報を求める

ここでは、HTML5のGeolocation APIを使い、一度だけ位置情報を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

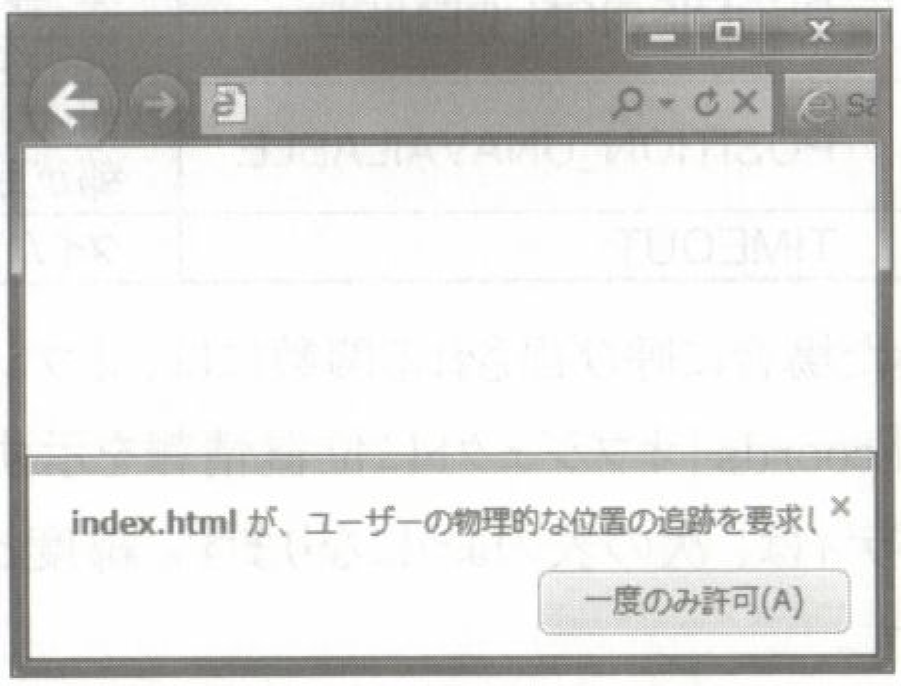
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Geolocation APIが使えるか調べる
  if (!navigator.geolocation){
    ele.innerHTML = "Geolocation APIは使えません";
    return;
  }
  // 一度だけ位置情報を取得する
  navigator.geolocation.getCurrentPosition(function(position){
    // 緯度を求める
    var lat = position.coords.latitude;
    // 経度を求める
    var lon = position.coords.longitude;
    // 緯度経度の誤差を求める
    var acc = position.coords.accuracy;
    // 高度を求める
    var alt = position.coords.altitude;
    // 高度の誤差を求める
    var accAlt = position.coords.altitudeAccuracy;
    // 結果を出力
    ele.innerHTML = "緯度 :"+lat+"<br>";
    ele.innerHTML += "経度 :"+lon+"<br>";
    ele.innerHTML += "緯度経度の誤差 :"+acc+"m<br>";
    ele.innerHTML += "高度 :"+alt+"<br>";
    ele.innerHTML += "高度の誤差 :"+accAlt+"<br>";
```

```
    },function(err){
      // エラーの場合。エラーコードを出力
      ele.innerHTML = "エラー:"+err.code;
    },{
      // 詳細な位置を返すように設定
      enableHighAccuracy : true
    });
  }, false);
```

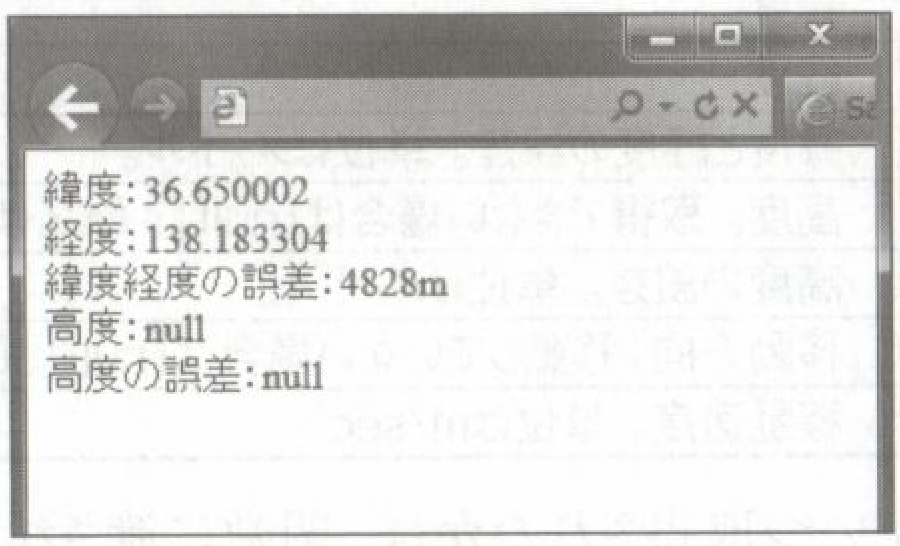
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示すると、位置情報を取得してもよいかどうかの確認が行われます。ここで拒否すると、位置情報は取得できません。



許可すると、取得した位置情報が表示されます。



ONEPOINT 位置情報を一度だけ取得するにはGeolocation APIの「getCurrentPosition()」メソッドを使う

位置情報を一度だけ取得するには、Geolocation APIの「getCurrentPosition()」メソッドを使います。「getCurrentPosition()」メソッドの最初のパラメータには、位置情報の取得に成功した場合に呼び出される関数を指定します。2番目のパラメータには、位置情報が取得できなかった場合、つまり、エラー時の処理を行う関数を指定します。2番目のパラメータは必須ではないので省略することができます。3番目のパラメータは、位置情報の取得に関するオプションを設定します。指定できるオプションは、次の表のようになります。

オプション	説明
enableHighAccuracy	「true」を指定すると、詳細な位置を返すようにする。ただし、スマートフォンではバッテリーの消耗が激しくなる
timeout	タイムアウトまでの時間をミリ秒で指定する
maximumAge	取得した位置情報を何ミリ秒有効にするか指定する。「0」を指定すると、常に最新の位置情報になる。ただし、スマートフォンではバッテリーの消耗が激しくなる。また、パソコンの場合は「0」を指定しても最新の情報が返されない場合がある

位置情報を得るには、ユーザーの許可が必要になります。ユーザーの許可が得られない場合は、エラーになります。他にも、位置情報が取得できない、時間切れ(タイムアウト)の場合は、次の表に示すエラーが返されます。このエラーコードは、エラー発生時に呼び出される関数に渡されるオブジェクトの「code」プロパティに格納されています。また、エラーメッセージは「message」プロパティに格納されています。

エラーコード値	キーワード値	説明
1	PERMISSION_DENIED	ユーザーが位置情報の提供を拒否した
2	POSITION_UNAVAILABLE	何らかの原因で位置情報が取得できなかった
3	TIMEOUT	タイムアウトになった

位置情報が取得できた場合に呼び出される関数には、オブジェクトが渡されます。このオブジェクト内にある「coords」オブジェクトに位置情報を示すプロパティがあります。位置情報を示すプロパティは、次の表のようになります。緯度と経度は、世界測地系WGS84に準拠しています。

プロパティ	説明
latitude	緯度
longitude	経度
altitude	緯度と経度の誤差。単位はメートル。
accuracy	高度。取得できない場合は「null」、もしくは、「undefined」
altitudeAccuracy	高度の誤差。単位はメートル。
heading	移動方向。移動していない場合は「null」、もしくは、「undefined」
speed	移動速度。単位はm/sec

これらの位置情報がいつ取得されたかは、関数に渡されたオブジェクトの「timestamp」プロパティで読み出すことができます。「timestamp」プロパティには、1970年1月1日午前0時からのミリ秒が入っています。

関連項目 ▶▶▶

● 定期的に位置情報を取得・停止する ……………………………………………… p.735

SECTION-310

定期的に位置情報を取得・停止する

ここでは、HTML5のGeolocation APIを使い、定期的に位置情報を求める方法と、定期的な監視を解除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

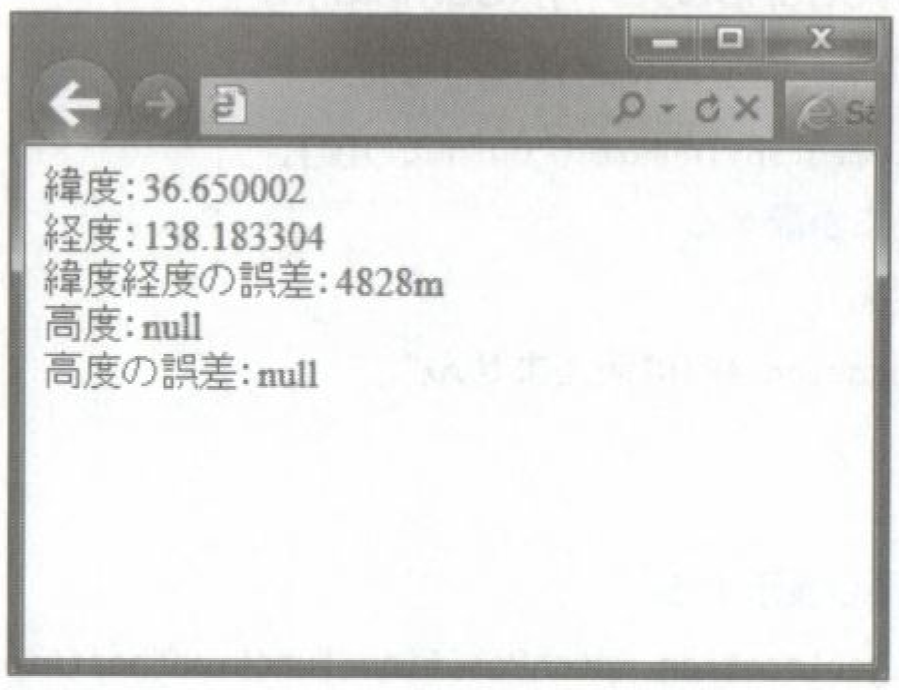
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Geolocation APIが使えるか調べる
  if (!navigator.geolocation){
    ele.innerHTML = "Geolocation APIは使えません";
    return;
  }
  // 定期的に位置情報を取得し表示する
  var watchID = navigator.geolocation.watchPosition(function(position){
    // 緯度を求める
    var lat = position.coords.latitude;
    // 経度を求める
    var lon = position.coords.longitude;
    // 緯度経度の誤差を求める
    var acc = position.coords.accuracy;
    // 高度を求める
    var alt = position.coords.altitude;
    // 高度の誤差を求める
    var accAlt = position.coords.altitudeAccuracy;
    // 結果を出力
    ele.innerHTML = "緯度 :"+lat+"<br>";
    ele.innerHTML += "経度 :"+lon+"<br>";
    ele.innerHTML += "緯度経度の誤差 :"+acc+"m<br>";
    ele.innerHTML += "高度 :"+alt+"<br>";
    ele.innerHTML += "高度の誤差 :"+accAlt+"<br>";
```

```
  },function(err){
    // エラーの場合。エラーコードを出力
    ele.innerHTML = "エラー:"+err.code;
  },{
    // 詳細な位置を返すように設定
    enableHighAccuracy : true
  });
  // 10秒経過したら取得を停止する
  setTimeout(function(){
    navigator.geolocation.clearWatch(watchID);
    ele.innerHTML += "位置情報の取得を停止";
  }, 1000*10);
}, false);
```

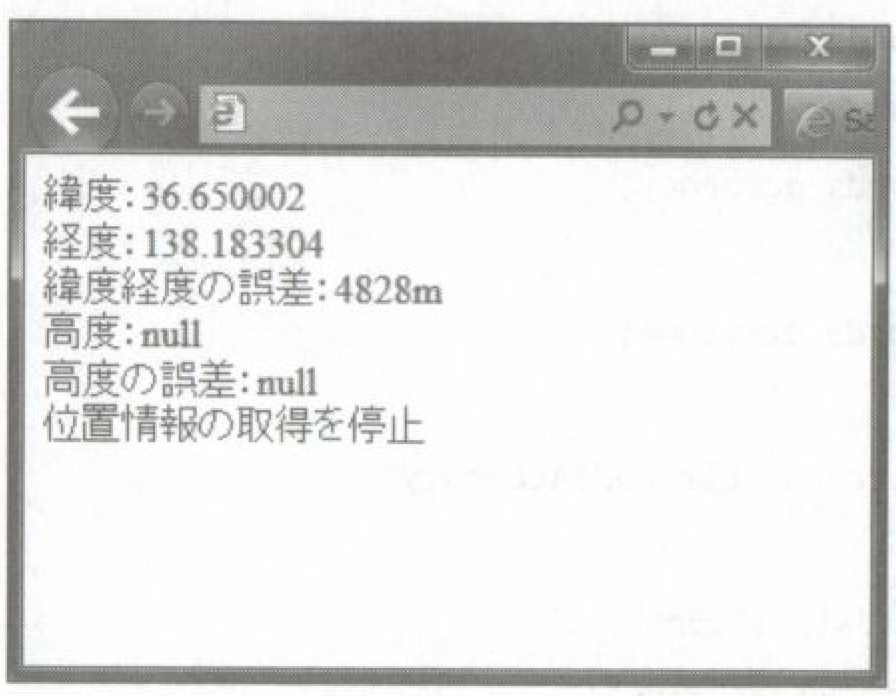
HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示すると、位置情報が表示されます。なお、この前に位置情報の取得を行ってもよいかどうかの確認メッセージが表示されることがあります。



10秒間だけ位置情報を取得し、10秒経過したら取得を停止して、その旨のメッセージが表示されます。



ONEPOINT **定期的に位置情報を取得するには「watchPosition()」メソッド、停止するには「clearWatch()」メソッドを使う**

定期的に位置情報を取得するには、「watchPosition()」メソッドを使います。位置情報を取得するメソッドとしては「getCurrentPosition()」メソッドがありますが、状況によっては位置情報が取得できない場合があります(地下やトンネル内など)。このような場合は、「watchPosition()」メソッドの方が向いています。ただし、「watchPositon()」メソッドは定期的に位置情報を取得するため、スマートフォンではバッテリーの消耗が激しくなります。

そこで、一定期間だけ位置情報を取得、もしくは、期待する精度の位置情報が取得できたら、監視を停止する方法があります。「watchPosition()」メソッドの動作を停止するには、「clearWatch()」メソッドを使います。パラメータには、「watchPosition()」メソッドが返すタイマーIDを指定します。

COLUMN **機能が利用できるか調べるには**

ブラウザの進歩は非常に速く、どの機能が使えるのか把握するのが難しくなっています。そのような場合に便利なのがModernizrです。Modernizrを使えば、特定の機能が利用できるかどうかを手軽に判別することができます。Modernizrについては、次のページを参照してください。

- Modernizr: the feature detection library for HTML5/CSS3

  URL http://modernizr.com/

関連項目 ▶▶▶

● 一度だけ位置情報を求める……………………………………………………………p.732

Web Storageにデータを設定する

ここでは、Web Storageにデータを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

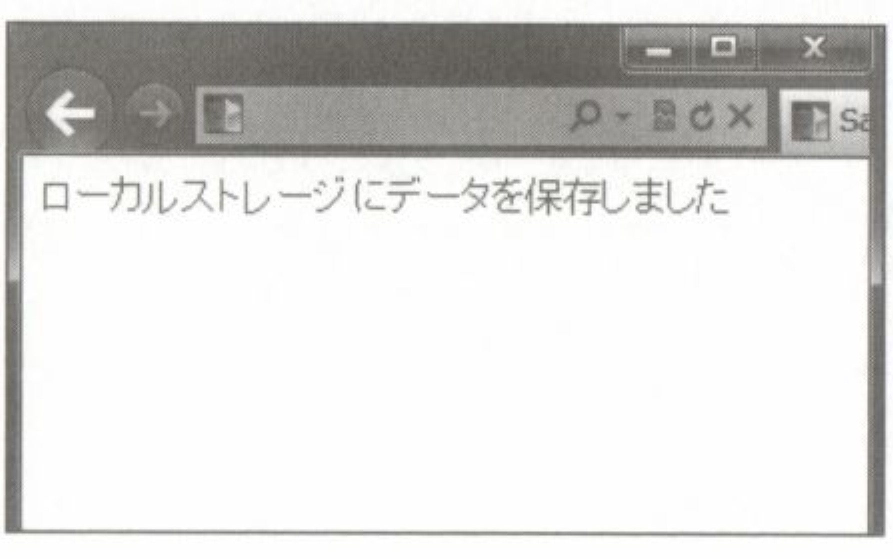
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ローカルストレージが使えるか調べる
  if (!localStorage){
    ele.innerHTML = "ローカルストレージは使えません";
    return;
  }
  localStorage.setItem("C&R", "研究所");
  localStorage.book = "JavaScript";
  localStorage["price"] = 3980;
  ele.innerHTML = "ローカルストレージにデータを保存しました";
}, false);
```

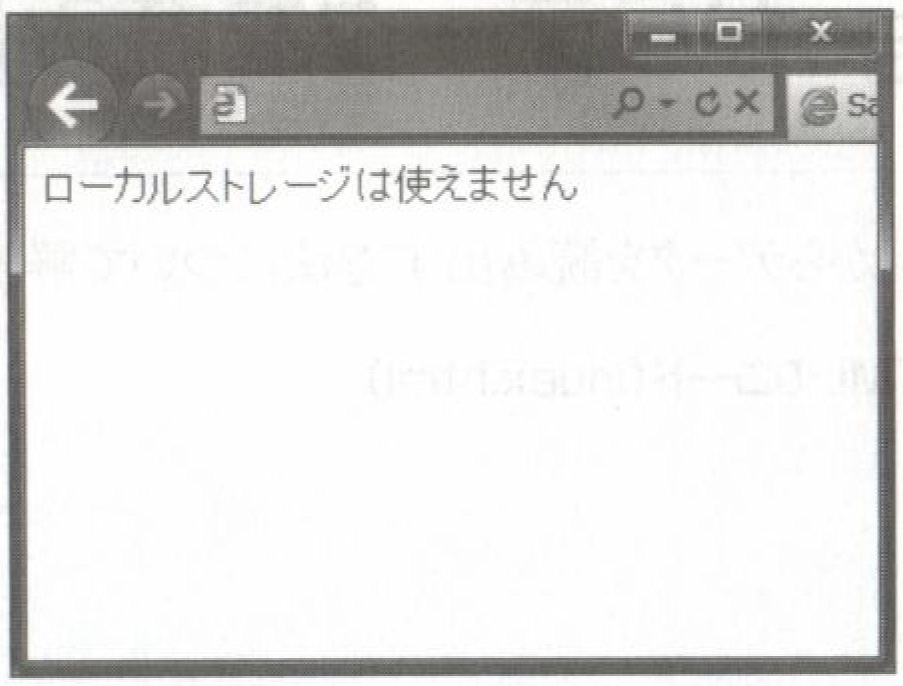
HINT

IE9〜10では、ローカル上ではローカルストレージを利用することができません。このため、「localStorage」は「undefined」となります。

ブラウザでindex.htmlを表示し、ローカルストレージにデータが保存されると、保存した旨のメッセージが表示されます。



ローカルストレージが使えない場合はデータは保存されず、メッセージだけが表示されます。



ONEPOINT ストレージを利用するには「sessionStorage」や「localStorage」を使う

Web Storageには、2種類あります。1つは、同じセッション間でデータを共有できる「sessionStorage」です。もう1つは、ローカルディスク上にデータを保存できる「local Storage」です。いずれも保存容量に制限がありますが、設定や実行環境によって異なります。ストレージ容量としては5MBが目安となっています。

ストレージに保存できるデータは同一オリジン(プロトコル、ポート名、ドメイン名が同一)で共有されます。オリジンが異なると、保存領域は別に割り当てられます。異なるオリジンのストレージにはアクセスすることはできません。

ストレージにデータを設定するには、3つの方法があります。

1つ目は、「setItem()」メソッドを使う方法です。最初のパラメータにキー名、2番目のパラメータに保存するデータを指定します。2つ目は、「localStorage["キー名"]=データ」のように、配列形式を利用する方法です。3つ目は、「localStorage.プロパティ名 = データ」のように、JavaScriptのオブジェクト+プロパティ名の形式で指定する方法です。3つめの指定方法では、キー名には特殊記号や空白を含めることはできません。

関連項目 ▶▶▶

- Web Storageからデータを読み出す …… p.740
- Web Storage内のすべてのキーと値を取得する …… p.742
- Web Storageの特定のデータを削除する …… p.744
- Web Storageのすべてのデータを削除する …… p.746
- ストレージ内容が変更された際のイベントを設定する …… p.748
- Indexed Databaseにデータを書き込む …… p.841

Web Storageからデータを読み出す

ここでは、Web Storageからデータを読み出す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

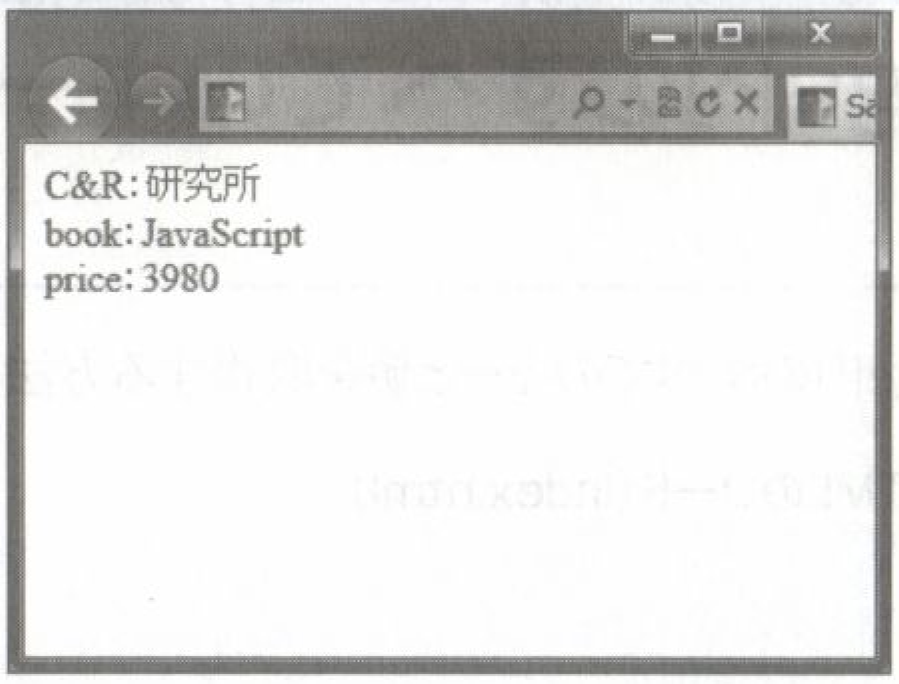
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ローカルストレージからC&Rのキーの値を読み出す
  var myData1 = localStorage.getItem("C&R");
  if (!myData1){ myData1 = "C&Rキーがありません"; }
  // ローカルストレージからbookのキーの値を読み出す
  var myData2 = localStorage.book;
  if (!myData2){ myData2 = "bookキーがありません"; }
  // ローカルストレージからbookのキーの値を読み出す
  var myData3 = localStorage["price"];
  if (!myData3){ myData3 = "priceキーがありません"; }
  // 読み出した結果を出力
  ele.innerHTML = "C&R:"+myData1;
  ele.innerHTML += "<br>book:"+myData2;
  ele.innerHTML += "<br>price:"+myData3;
}, false);
```

HINT

IE9〜10では、ローカル上ではローカルストレージを利用することができません。このため、ローカル上ではデータを読み出すことはできません。

ブラウザでindex.htmlを表示し、ローカルストレージからデータを読み出した結果は、次のようになります。これは、キーが存在する場合です。



ローカルストレージにデータがない場合は、次のように表示されます。



ONEPOINT ストレージから読み出すには「getItem()」メソッドを使う

Web Storageに保存されたデータを読み出すには、「getItem()」メソッドを使い、パラメータにはキーを指定します。該当するキーがない場合、「getItem()」は「null」を返します(「undefined」ではない)。

保存されたデータを読み出す方法には、「getItem()」メソッドを使う以外にもあります。キー名に空白や特殊記号を含んでいない場合は「localStorage.キー名」や「sessionStorage.キー名」のように、オブジェクトのプロパティとして指定し読み出すことができます。キーが存在しない場合は、「null」、または、「undefined」になります(ブラウザによって返される結果が異なる)。

また、配列(ハッシュ)形式でも指定することができます。この場合は、「localStorage["キー名"]」「sessionStorage["キー名"]」としてデータを読み出すことができます。配列形式で指定した場合、該当するキーがないと「null」、または、「undefined」を返します。

関連項目 ▶▶▶

- Web Storageにデータを設定する ……… p.738
- Web Storage内のすべてのキーと値を取得する ……… p.742
- Web Storageの特定のデータを削除する ……… p.744
- Web Storageのすべてのデータを削除する ……… p.746
- ストレージ内容が変更された際のイベントを設定する ……… p.748
- Indexed Databaseからデータを読み出す ……… p.845

SECTION-313

Web Storage内のすべてのキーと値を取得する

ここでは、Web Storage内のすべてのキーと値を取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

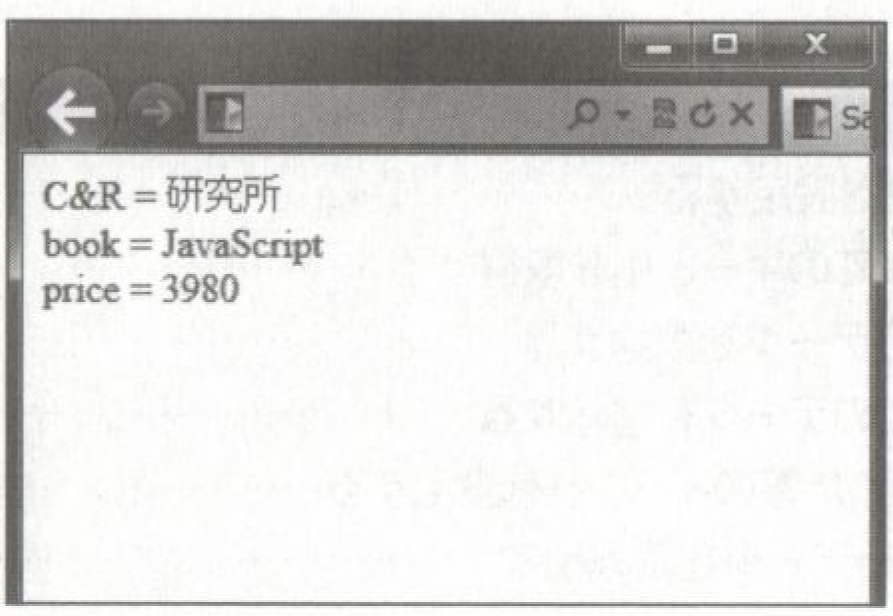
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ローカルストレージからすべてのキーを読み出して出力する
  for(var i=0; i<localStorage.length; i++){
    // キーを読み出す
    var tempKey = localStorage.key(i);
    var data = localStorage.getItem(tempKey);
    // キーと値を出力
    ele.innerHTML += tempKey+" = "+data+"<br>";
  }
}, false);
```

HINT

IE9～10では、ローカル上ではローカルストレージを利用することができません。このため、ローカル上ではデータを読み出すことはできません。

ブラウザでindex.htmlを表示すると、ローカルストレージ内のキーを読み出し、そのキーを元に読み出した内容がまとめて表示されます。



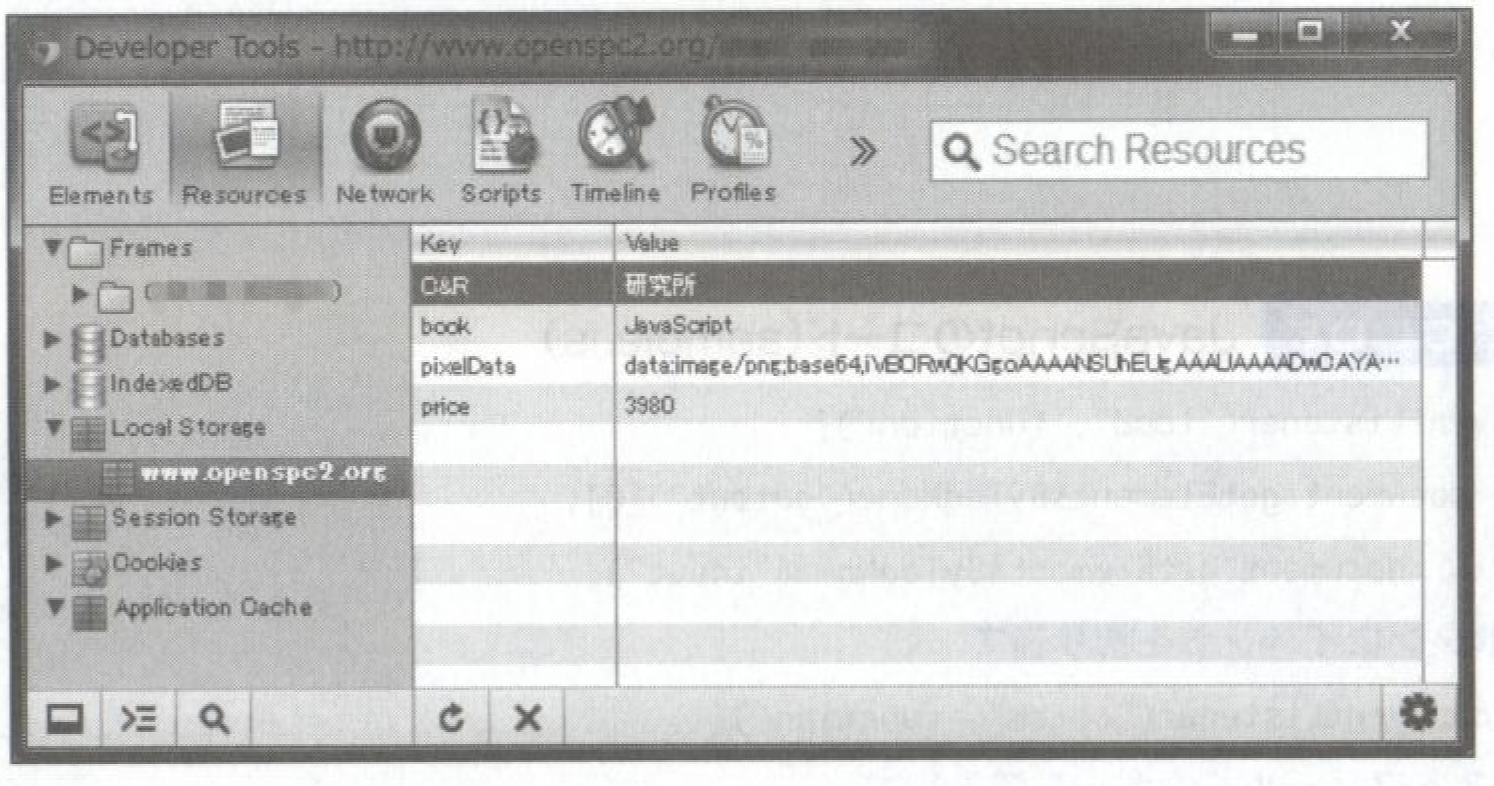
ONEPOINT キーを取得するには「key()」メソッドを使う

Web Storageは、Key Value Store方式でデータを格納しています。このため、キーがわからないとデータを取り出すことができません。Web Storageでは、同じオリジンであれば、「key()」メソッドを使ってすべてのキーを取得することができます。キーを取得してしまえば、後は自由に内容を読み出したり変更することができます。ストレージに、どのくらいのキーがあるかは、「localStorage.length」「sessionStorage.length」のように、「length」プロパティで求めることができます。

「key()」メソッドは同じオリジン内のすべてのキーを読み出すことができるため、同じオリジンで複数のユーザーが共同で利用しているような場合には注意が必要です。

COLUMN Web Storageに保存されている内容を調べる

ブラウザに用意されているデバッグツールを使うと、次の図のように、Web Storageに保存されているデータを確認したり、削除・変更することができます。なお、IE9の開発者ツールには、Web Storageを処理する機能はありません。



関連項目 ▶▶▶

SECTION-314

Web Storageの特定のデータを削除する

ここでは、LocalStorage、および、SessionStorage内にある特定のデータを削除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
    キー名：<input type="text" value="price"><br>
      <input type="button" value="該当するキーとデータを消去">
    </form>
    <output></output>
  </body>
</html>
```

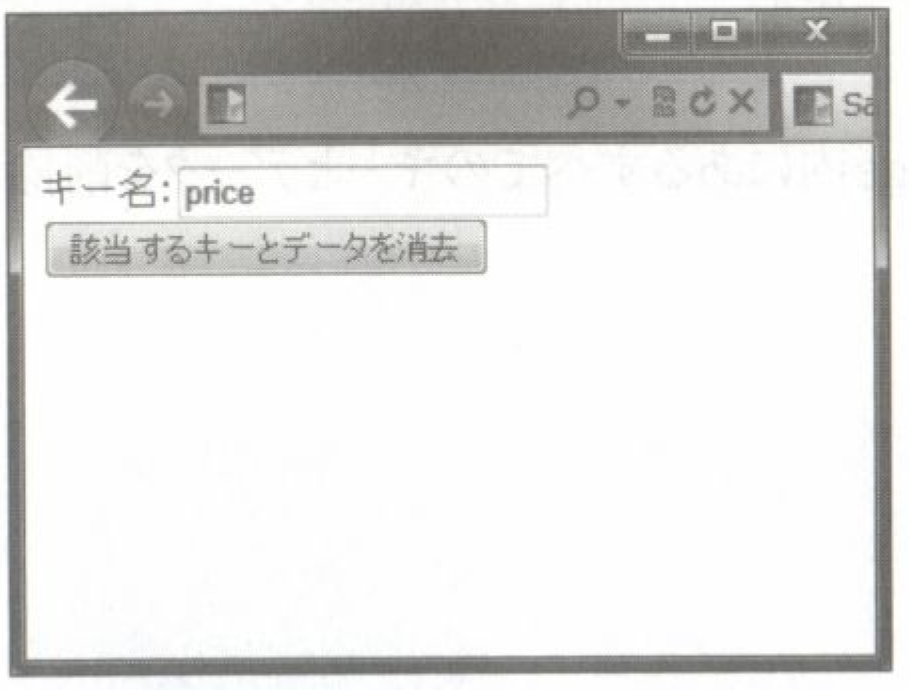
SAMPLE CODE JavaScriptのコード(sample.js)

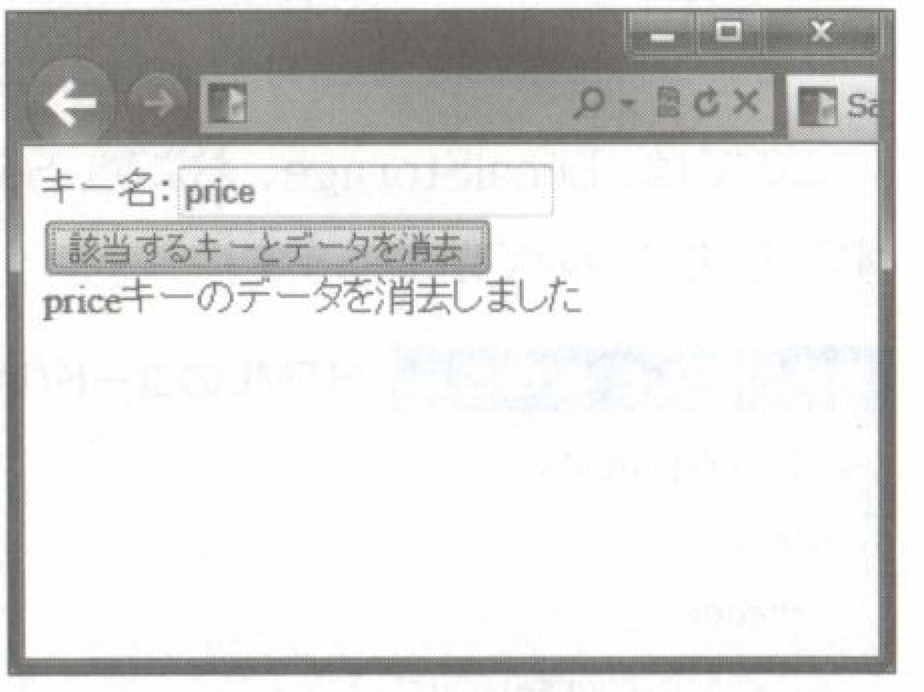
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var input =  document.getElementsByTagName("input");
  // inputボタンにイベントを割り当て
  input[1].addEventListener("click", function(){
    // テキストフィールドのキーを読み出す
    var tempKey = input[0].value;
    // キーとデータを削除
    localStorage.removeItem(tempKey);
    // メッセージを出力
    ele.innerHTML = tempKey+"キーのデータを消去しました";
  }, false);
}, false);
```

HINT

IE9～10では、ローカル上ではローカルストレージを利用することができません。

ブラウザでindex.htmlを表示し、削除するデータに対応するキーを入力してボタンをクリックすると、ローカルストレージの該当するキーのデータが消去されます。





ONEPOINT 特定のデータを消去するには「removeItem()」メソッドを使う

ストレージに保存された特定のデータを消去するには、「removeItem()」メソッドを使います。「removeItem()」メソッドのパラメータには削除するキーを指定します。「removeItem()」は、対応するキーがない場合でもエラーにはなりません(処理が行われず無視される)。

なお、ブラウザのデバッグツールを使って、ローカルストレージに保存されたデータを削除することもできます。

関連項目 ▶▶▶

- Web Storageにデータを設定する ………… p.738
- Web Storageからデータを読み出す ………… p.740
- Web Storage内のすべてのキーと値を取得する ………… p.742
- Web Storageのすべてのデータを削除する ………… p.746
- ストレージ内容が変更された際のイベントを設定する ………… p.748

SECTION-315

Web Storageのすべてのデータを削除する

ここでは、LocalStorage、および、SessionStorage内にあるすべてのキーとデータを削除する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <button>LocalStorage内のデータを全て消去する</button><br>
    <output></output>
  </body>
</html>
```

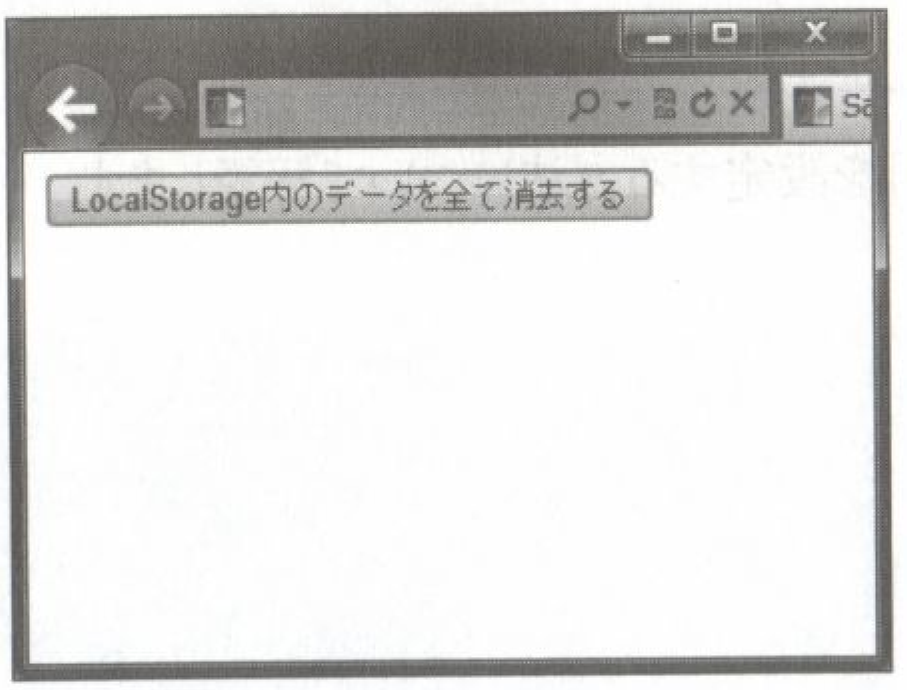
SAMPLE CODE JavaScriptのコード(sample.js)

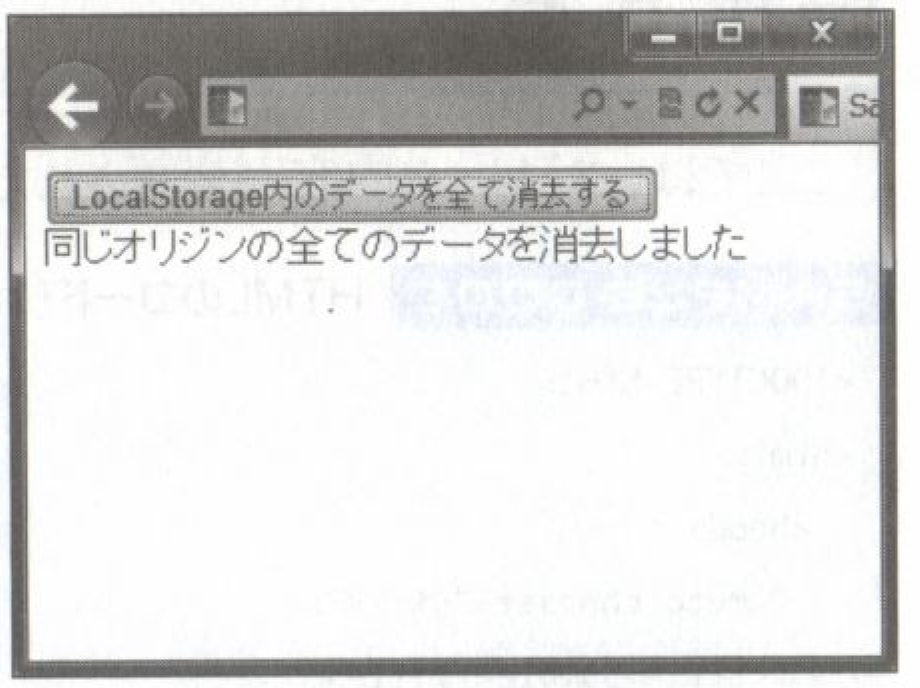
```
window.addEventListener("load", function(){
  // ボタンにイベントを割り当て
  var clearButton =  document.getElementsByTagName("button")[0];
  clearButton.addEventListener("click", function(){
    // LocalStorage内のデータを消去
    localStorage.clear();
    // メッセージを表示
    var ele = document.getElementsByTagName("output")[0];
    ele.innerHTML = "同じオリジンの全てのデータを消去しました";
  }, false);
}, false);
```

HINT

IE9～10では、ローカル上ではローカルストレージを利用することができません。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、ローカルストレージの同じオリジンのすべてのデータが消去されます。





ONEPOINT 全データを消去するには「clear()」メソッドを使う

ストレージに保存されたデータを一括して消去するには、「clear()」メソッドを使います。ただし、「clear()」メソッドで消去されるのは、同じオリジン内のキーとデータだけです。異なるオリジンのデータが消去されることはありません。また、同じオリジンでもlocalStorageとsessionStorageは別々に管理されているため、「localStorage.clear()」としても、同じオリジンのsessionStorageの内容は消去されません。

関連項目 ▶▶▶

- Web Storageにデータを設定する …… p.738
- Web Storageからデータを読み出す …… p.740
- Web Storage内のすべてのキーと値を取得する …… p.742
- Web Storageの特定のデータを削除する …… p.744
- ストレージ内容が変更された際のイベントを設定する …… p.748

SECTION-316

ストレージ内容が変更された際のイベントを設定する

ここでは、ストレージ内容が変更された際のイベントを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <button>現在時刻をストレージに保存</button><br>
    <output></output>
  </body>
</html>
```

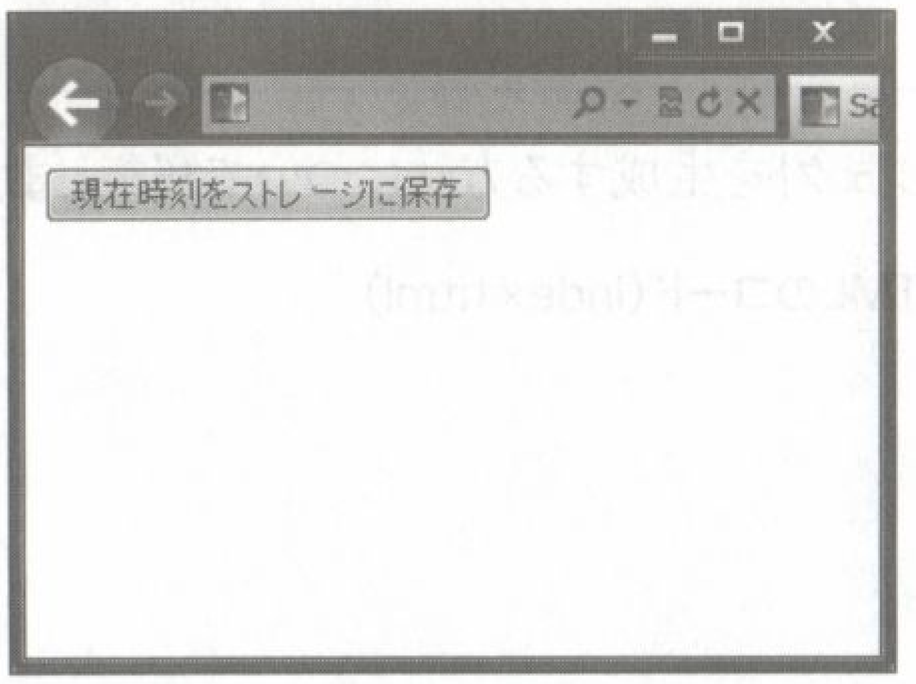
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var btn =  document.getElementsByTagName("button")[0];
  // inputボタンにイベントを割り当て
  btn.addEventListener("click", function(){
    // キーとデータを削除
    localStorage.setItem("xDay", new Date());
  }, false);
  // windowオブジェクトにstorageイベントを割り当て
  window.addEventListener("storage", function(evt){
    var tempKey = evt.key;
    ele.innerHTML = tempKey+"キーのデータが更新されました<br>";
    ele.innerHTML += evt.oldValue+"<br>↓<br>"+evt.newValue;
  }, false);
}, false);
```

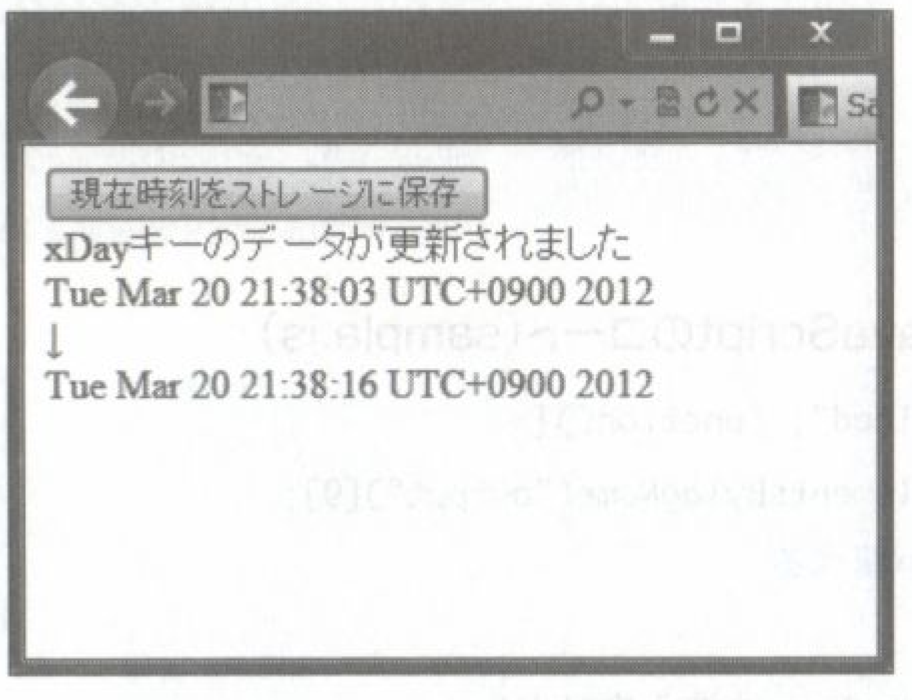
HINT

IE9～10では、ローカル上ではローカルストレージを利用することができません。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、現在の時刻がローカルストレージに保存されます。このとき、「xDay」というキーが使われます。



データが保存されると「storage」イベントが発生し、変更前のデータと変更後のデータも表示されます。



ONEPOINT データ保存時の処理を設定するには「storage」イベントを使う

Web Storageにデータが保存されると、「storage」イベントが発生します。「storage」イベントが発生時イベントハンドラに渡されるイベントオブジェクトには、次のデータが渡されます。

プロパティ	説明
key	キー
oldValue	変更前の値
newValue	変更後の値
url	URL
Storage	ストレージの種類

関連項目 ▶▶▶

- イベントハンドラを設定する …… p.396
- イベントリスナーを設定する …… p.398
- Web Storageにデータを設定する …… p.738
- Web Storageからデータを読み出す …… p.740
- Web Storage内のすべてのキーと値を取得する …… p.742
- Web Storageの特定のデータを削除する …… p.744
- Web Storageのすべてのデータを削除する …… p.746

SECTION-317

ワーカーオブジェクトを生成する

ここでは、ワーカーオブジェクトを生成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

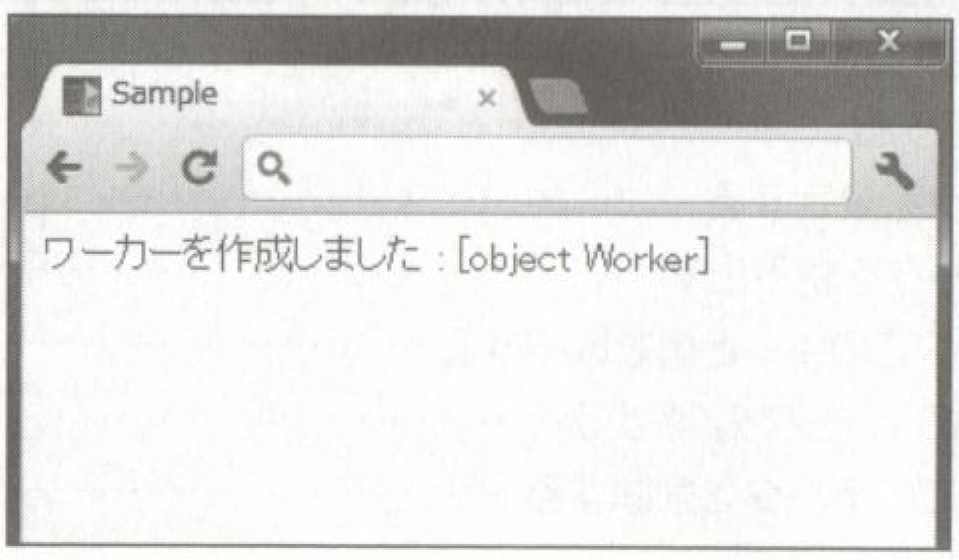
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Workersが使えるか調べる
  if (!window.Worker){
    ele.innerHTML = "Web Workersは使えません";
    return;
  }
  // ワーカーオブジェクトを作成
  var myWorker = new Worker("js/add.js");
  // 結果を出力
  ele.innerHTML = "ワーカーを作成しました : "+myWorker;
  // ワーカーオブジェクトを破棄
  myWorker = null;
}, false);
```

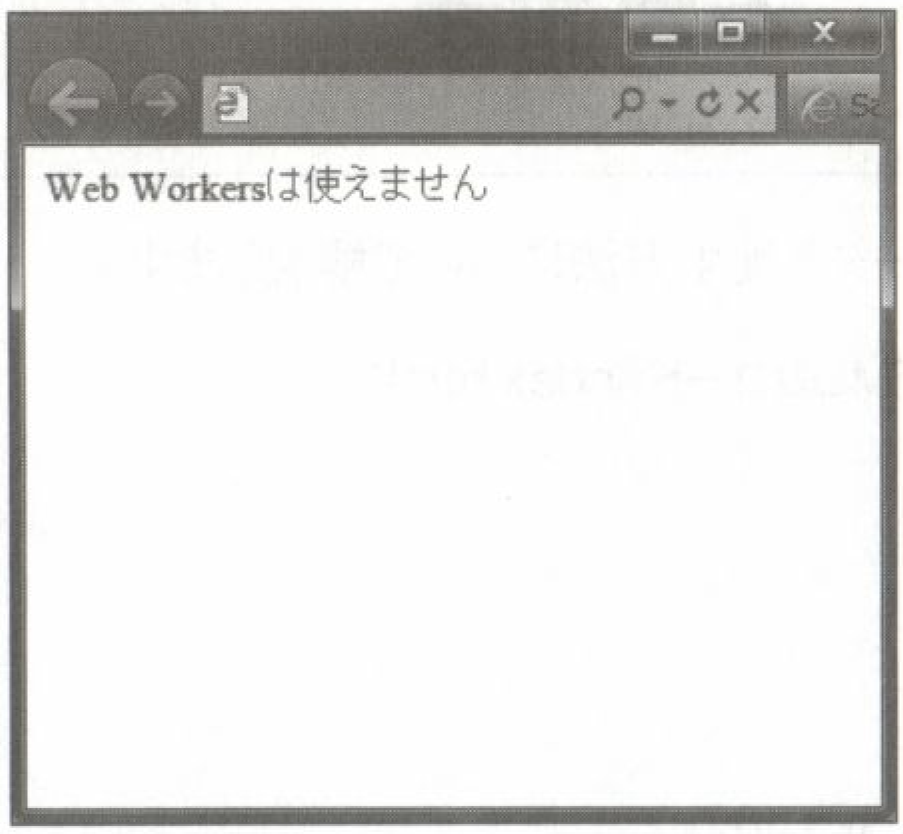
HINT

サンプルの動作を確認する際は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示すると、ワーカーオブジェクトが生成されます。



IE9ではWeb Workersは利用できないため、次のようなメッセージが表示されます。



ONEPOINT **ワーカーオブジェクトを作成するには「new Worker()」を使う**

JavaScriptは基本的に、同時に複数の処理を行うことができません。そこで、並列に処理を行うには、Web Workersを利用します。ワーカーオブジェクトを生成するには「new Worker()」を使い、パラメータに実行するスクリプトファイル名を指定します。指定されたスクリプトは、すぐに実行されます。

ただし、すぐに実行されると困る場合がほとんどです。そこで、752ページで説明する「postMessage()」を使ってワーカーと呼び出し側でデータのやり取りを行います。

ワーカー側のスクリプトには、制約があります。「window」オブジェクトにはアクセスできないため、ドキュメント操作はもちろん、Web Storageなどにもアクセスすることはできません。ただし、File APIは利用できるので、時間のかかるファイル処理をワーカー側で行うことはできます。

また、Android 4ではWeb Workersを利用することができますが、それ以前のAndroid 2ではJavaScript処理とUI処理は別々のスレッドで処理されます。このため、JavaScriptで時間のかかる処理を行っても、UI操作がまったくできなくなってしまうことはありません。

関連項目 ▶▶▶

- 非同期処理を行う……p.292
- ワーカーにデータを渡す……p.752
- ワーカー側でJavaScriptファイルを読み込む……p.754
- ワーカーを終了する……p.756

ワーカーにデータを渡す

ここでは、ワーカーにデータを渡す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Workersが使えるか調べる
  if (!window.Worker){
    ele.innerHTML = "Web Workersは使えません";
    return;
  }
  // ワーカーオブジェクトを作成
  var myWorker = new Worker("js/add.js");
  // 先にイベントハンドラを設定
  myWorker.onmessage = function(evt){
    ele.innerHTML = "合計結果: " + evt.data;
  }
  // ワーカーにメッセージを送信
  myWorker.postMessage({ start : 1, end: 10 });
}, false);
```

SAMPLE CODE JavaScriptのコード/ワーカー側のコード(add.js)

```
onmessage = function(evt){
  var startValue = evt.data.start;
  var endValue = evt.data.end;
  var total = 0;
  for(var i=startValue; i<=endValue; i++){
    total += i;
  }
  postMessage(total);
}
```

HINT

サンプルの動作を確認する際は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示すると、Web Workersで1〜10までの数値を合計した結果が返され、ページ上に表示されます。



ONEPOINT **ワーカーとやり取りするには「postMessage()」メソッドを使う**

ワーカーとデータのやり取りを行うには、「postMessage()」メソッドを使います。「postMessage()」メソッドは、ワーカー呼び出し側とワーカー側の両方で実装されています。つまり、相互にデータをやり取りすることができます。

「postMessage()」メソッドのパラメータには、JavaScriptのオブジェクトや数値などを指定できます。「postMessage()」メソッドでデータが渡されると、渡された側では「message」イベントが発生します。「onmessage」にイベントハンドラを設定したり、「addEventListener()」メソッドでイベントリスナーとして登録したりすることもできます。

イベントハンドラにはイベントオブジェクトが渡されます。イベントオブジェクトの「data」プロパティに、「postMessage()」メソッドで渡されたデータが入っています。

関連項目 ▶▶▶

- ワーカーオブジェクトを生成する ……… p.750
- ワーカー側でJavaScriptファイルを読み込む ……… p.754
- ワーカーを終了する ……… p.756

SECTION-319

ワーカー側でJavaScriptファイルを読み込む

ここでは、ワーカー側で複数のJavaScriptファイルを読み込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Workersが使えるか調べる
  if (!window.Worker){
    ele.innerHTML = "Web Workersは使えません";
    return;
  }
  // ワーカーオブジェクトを作成
  var myWorker = new Worker("js/main.js");
  // 先にイベントハンドラを設定
  myWorker.onmessage = function(evt){
    ele.innerHTML = "1〜100までの合計: " + evt.data;
  }
  // ワーカーにメッセージを送信
  myWorker.postMessage({ start : 1, end: 100 });
}, false);
```

SAMPLE CODE JavaScriptのコード/ワーカー側のコード(main.js)

```
importScripts("./add.js");
onmessage = function(evt){
  var startValue = evt.data.start;
  var endValue = evt.data.end;
  var total = addFunc(startValue, endValue);
  postMessage(total);
}
```

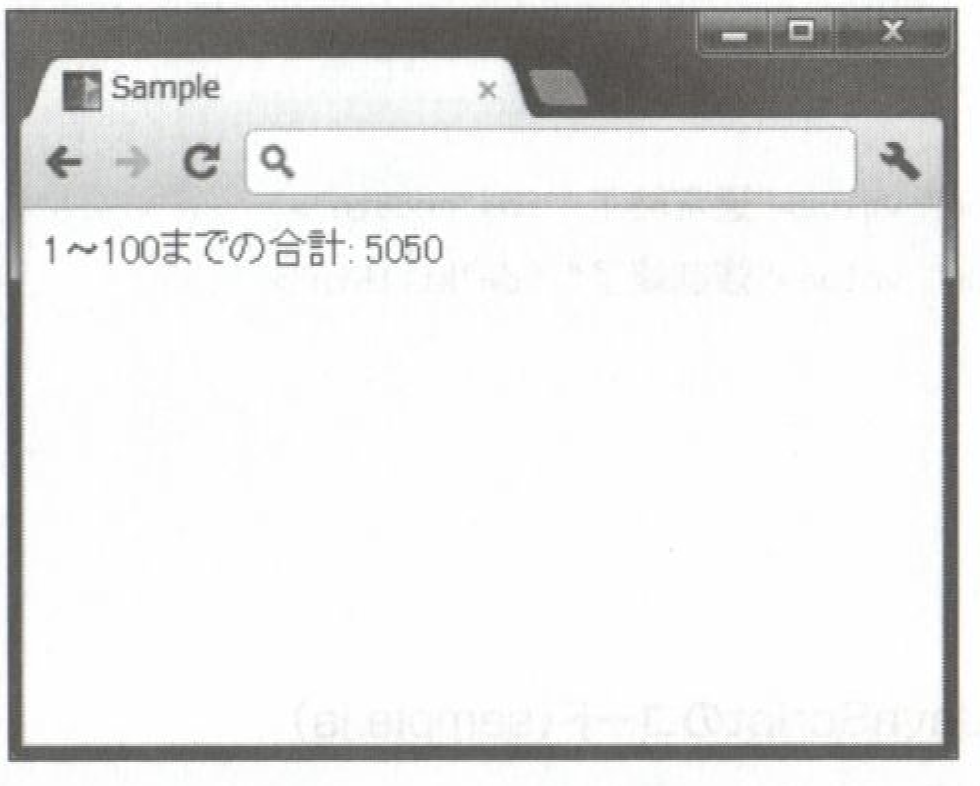
SAMPLE CODE JavaScriptのコード/ワーカー側のコード(add.js)

```
function addFunc(startValue, endValue){
  var total = 0;
  for(var i=startValue; i<=endValue; i++){
    total += i;
  }
  return total
}
```

HINT

サンプルの動作を確認する際は、ネットワーク環境(Webサーバー上)で確認してください。

ブラウザでindex.htmlを表示すると、ワーカー側で合計処理を行う関数が書かれたJavaScriptファイルが読み込まれ、実行した結果が表示されます。



ONEPOINT ワーカー側でJavaScriptファイルを読み込むには「importScripts()」メソッドを使う

ワーカー側では、「importScripts()」メソッドを使うことで、複数のJavaScriptファイルを読み込ませることができます。「importScripts()」は、パラメータに読み込ませるJavaScriptファイルのURLやパスを指定します。ファイルパスの場合、「importScripts()」メソッドを実行しているファイルのある場所が基準になります。

また、読み込ませるJavaScriptファイルはワーカー側で実行できるものでなければいけません。ファイルは指定した順番に読み込まれ、即時、実行されていきます。

関連項目 ▶▶▶

● ワーカーオブジェクトを生成する …… p.750
● ワーカーにデータを渡す …… p.752
● ワーカーを終了する …… p.756

SECTION-320

ワーカーを終了する

ワーカーは、呼び出し側で終了させる方法とワーカー側で終了させる方法があります。ここでは、ワーカーを終了する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="通常終了" id="endBtn">
      <input type="button" value="強制終了" id="killBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Workersが使えるか調べる
  if (!window.Worker){
    ele.innerHTML = "Web Workersは使えません";
    return;
  }
  // ワーカーオブジェクトを作成
  var myWorker = new Worker("js/add.js");
  // 先にイベントハンドラを設定
  myWorker.onmessage = function(evt){
    ele.innerHTML = evt.data;
    // ワーカーを呼び出す
    myWorker.postMessage(1);
  }
  // 通常終了ボタンにイベントを設定
  document.getElementById("endBtn").addEventListener("click", function(){
    myWorker.postMessage(false);
  }, false);
  // 強制終了ボタンにイベントを設定
  document.getElementById("killBtn").addEventListener("click", function(){
```

```
    // 強制終了
    myWorker.terminate();
  }, false);
  // ワーカーにメッセージを送信
  myWorker.postMessage(1);
}, false);
```

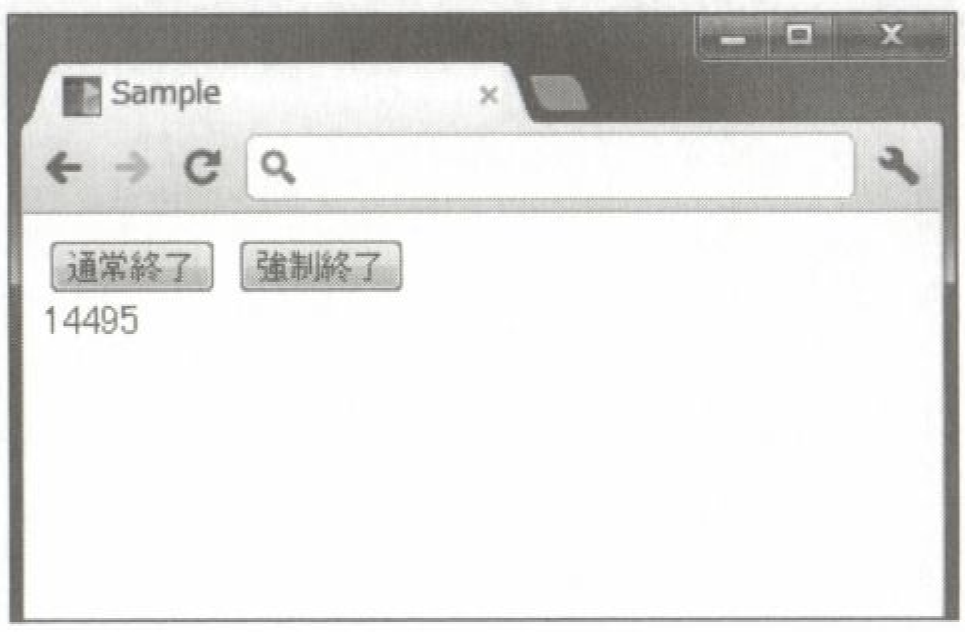
SAMPLE CODE JavaScriptのコード/ワーカー側のコード(add.js)

```
var total = 0;
onmessage = function(evt){
  // 終了コマンドかどうか調べる(falseを終了コマンドとしている)
  if (evt.data === false){
    postMessage("正常終了。合計:"+total);
    // ワーカーの処理を終了
    close();
  }else{
    // 数値の場合は計算を継続
    total += parseInt(evt.data);
    postMessage(total);
  }
}
```

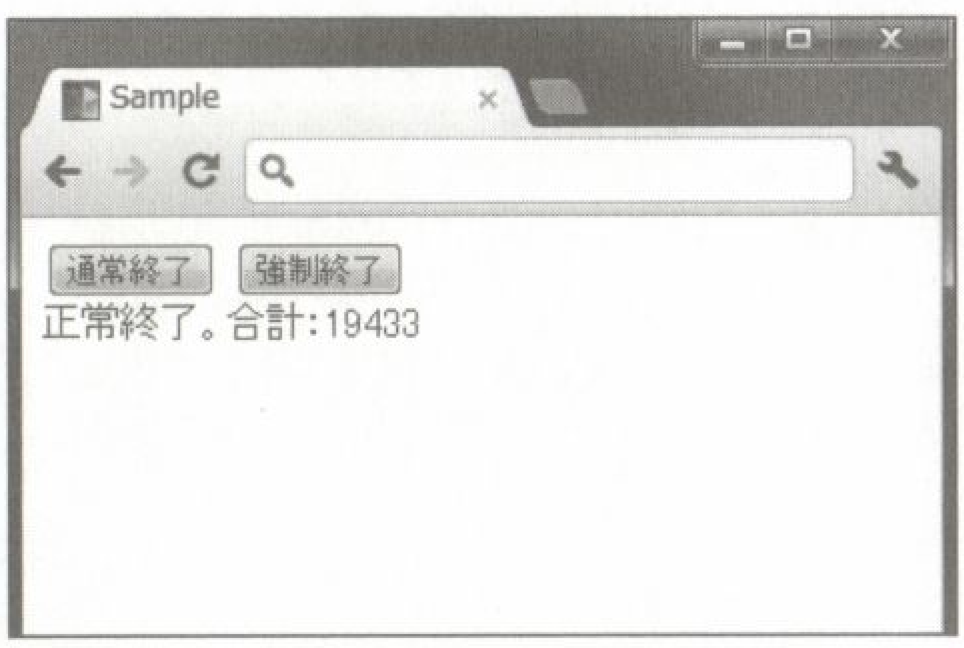
HINT

サンプルの動作を確認する際は、ネットワーク環境(Webサーバー上)で確認してください。

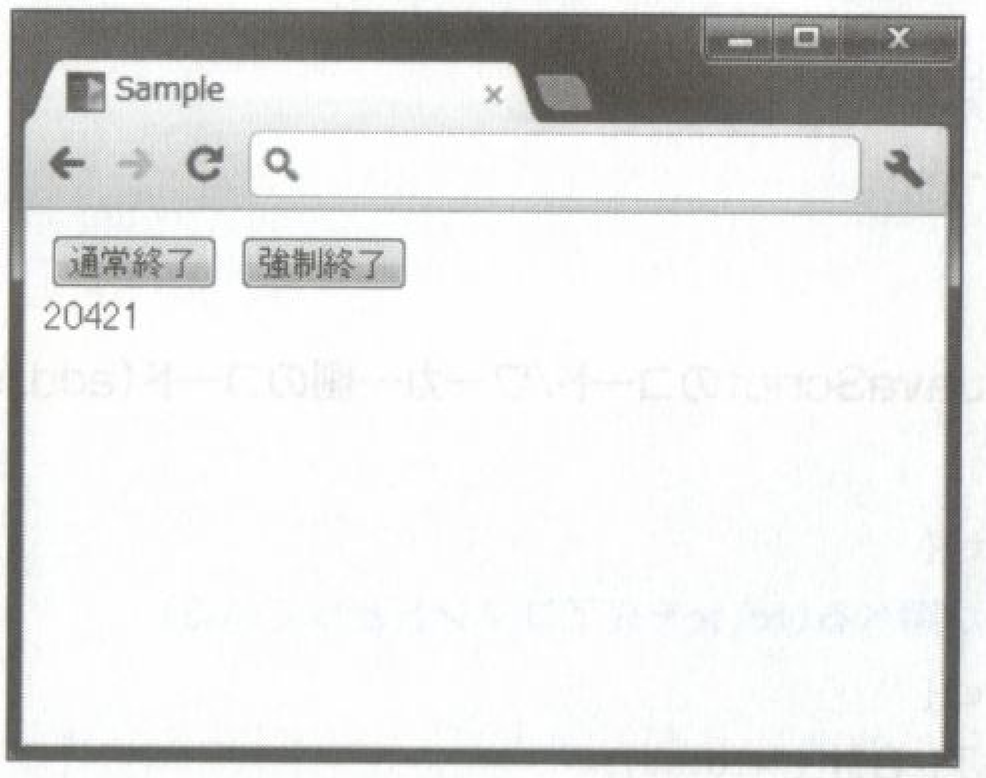
ブラウザでindex.htmlを表示すると、リアルタイムに数値が1ずつ加算されます。



通常終了のボタンをクリックすると、ワーカーの処理が終了し、メッセージと合計結果が表示されます。



強制終了のボタンをクリックすると、ワーカー側の処理が強制的に止まり、数値の加算も中断されます。



ONEPOINT

ワーカーを終了するには「terminate()」メソッドや「close()」メソッドを使う

バックグランドで動作しているワーカーの処理を停止させるには、「terminate()」メソッドを使います。「terminate()」は、動作しているワーカーを強制的に停止させます。また、ワーカー側で停止させる場合は、「close()」メソッドを使います。

ワーカーには、処理を一時停止させるメソッドはありません。一時的に停止させる場合は、「postMessage()」メソッドを使って自前で実装する必要があります。

関連項目 ▶▶▶

- ワーカーオブジェクトを生成する …… p.750
- ワーカーにデータを渡す …… p.752
- ワーカー側でJavaScriptファイルを読み込む …… p.754

オフライン/オンライン状態を調べる

ここでは、ブラウザのネットワークへの接続状態がオフラインかオンラインかを調べる方法について解説します。

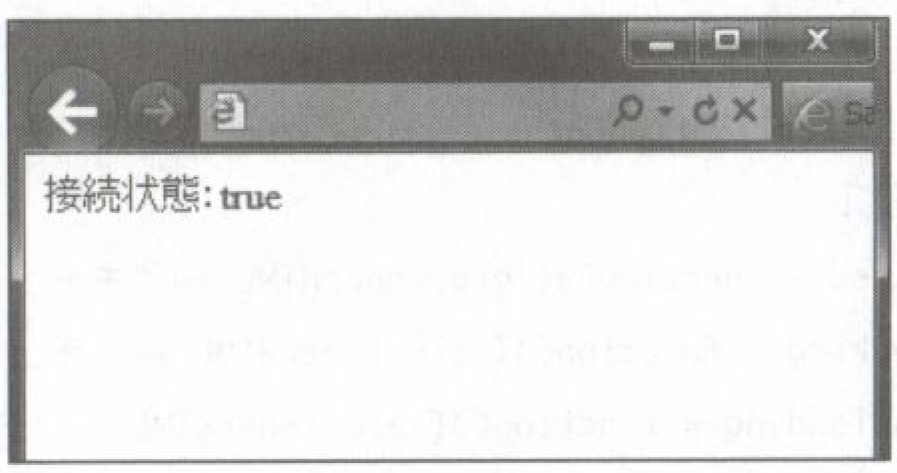
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // オンラインかオフラインか調べる
  var flag = navigator.onLine;
  // 結果を出力
  ele.innerHTML = "接続状態 :"+flag;
}, false);
```

ブラウザでindex.htmlを表示すると、接続状態が表示されます。



ONEPOINT オンラインかどうか調べるには「onLine」プロパティを使う

ブラウザがネットワーク回線に接続されているかどうかは、「navigator」オブジェクトの「onLine」プロパティで調べることができます。「onLine」プロパティが「true」ならネットワークに接続されていて、「false」の場合は接続されていないことになります。

Firefoxの場合は物理的にネットワークに接続されていても、ファイルメニューから「オフライン作業」を選択すると、「onLine」プロパティは「false」を返します。

アプリケーションキャッシュを更新する

ここでは、アプリケーションキャッシュを更新する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html manifest="money.appcache">
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="button" value="キャッシュ更新チェック" id="checkBtn">
      <input type="button" value="キャッシュを更新する" id="swapBtn">
    </form>
    <div><img src="images/sample.jpg" width="160" height="90"></div>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 更新チェックボタンにイベントを割り当て
  document.getElementById("checkBtn").addEventListener("click", function(){
    // キャッシュ更新
    applicationCache.update();
    // 結果を出力
    ele.innerHTML = "状態："+applicationCache.status+"<br>";
    // イベントハンドラを設定
    applicationCache.oncached = function(){ ele.innerHTML += "キャッシュ成功<br>"; }
    applicationCache.onchecking = function(){ ele.innerHTML += "チェック中...<br>"; }
    applicationCache.ondownloading = function(){ ele.innerHTML += "ダウンロード中...<br>"; }
    applicationCache.onprogress = function(){ ele.innerHTML += "キャッシュ更新中...<br>"; }
    applicationCache.onupdateready = function(){ ele.innerHTML += "更新準備完了<br>"; }
    applicationCache.onerror = function(){ ele.innerHTML += "エラー発生<br>"; }
  }, false);
  // 更新ボタンにイベントを割り当て
  document.getElementById("swapBtn").addEventListener("click", function(){
    if (applicationCache.status == applicationCache.UPDATEREADY){
      // キャッシュ更新
      applicationCache.swapCache();
```

01 02 03 04 05 06 07 08 09 10 11 12 CHAPTER 13 14 HTML5のその他のAPI

```
            // 結果を出力
            ele.innerHTML = "キャッシュを更新しました。3秒後にリロードします";
            // 3秒後にリロード
            setTimeout("location.reload()", 3*1000);
        }else{
            // 結果を出力
            ele.innerHTML = "キャッシュを更新できませんでした";
        }
    }, false);
}, false);
```

SAMPLE CODE アプリケーションキャッシュのコード(money.appcache)

```
CACHE MANIFEST
#ver 1.06

CACHE:
images/sample.jpg

NETWORK:
index.html
js/sample.js
```

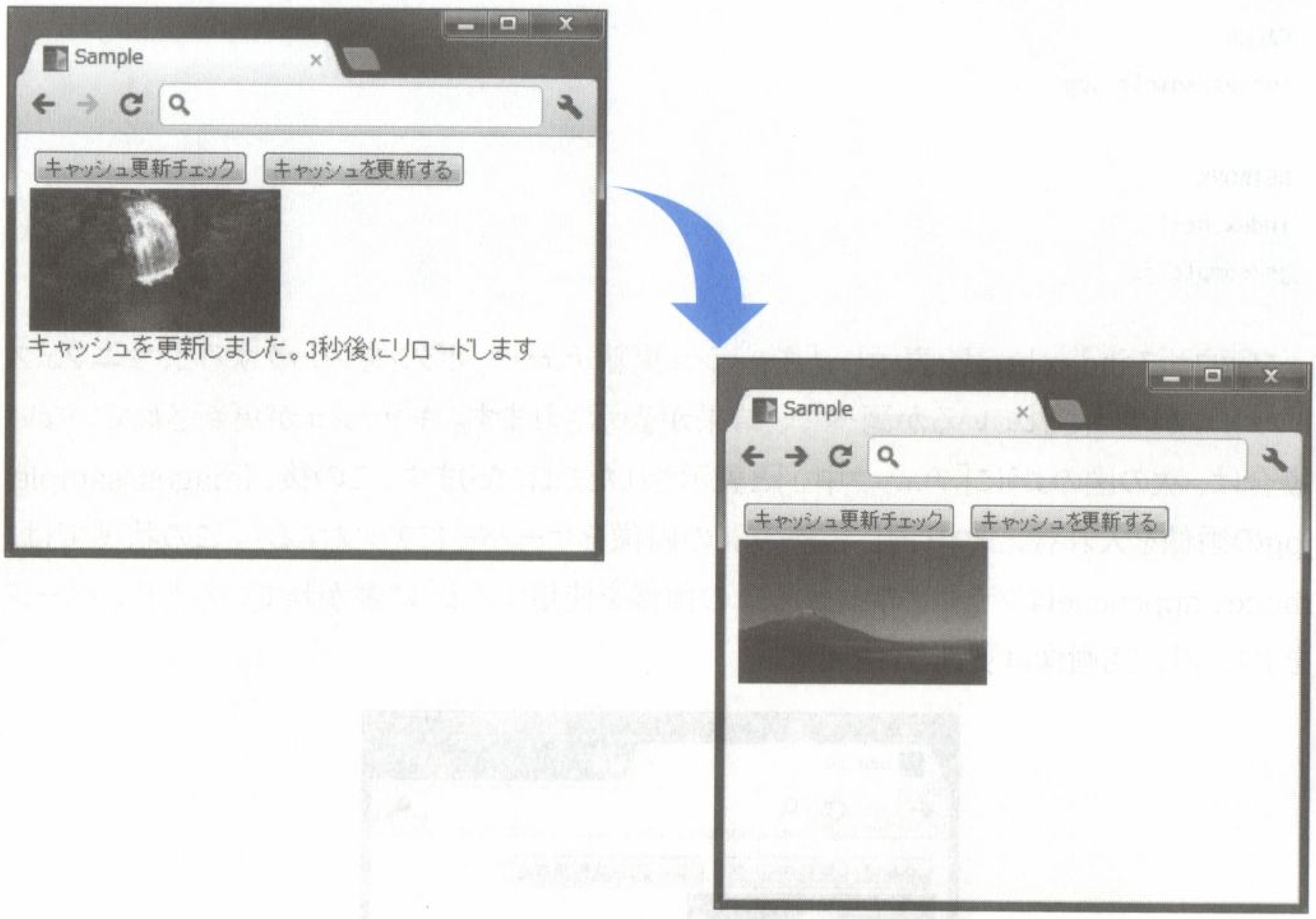
ブラウザでindex.htmlを表示し、「キャッシュ更新チェック」ボタンをクリックすると、マニフェストファイルが更新されているか調べて、結果が表示されます。キャッシュが更新されていない場合は、次の図のように「チェック中...」と表示されたままになります。この後、images/sample.jpgの画像を入れ替えます(同じ名前の別の画像をサーバーにアップする)。この状態では、money.appcacheはブラウザのキャッシュの画像を使用するように書かれているため、ページをリロードしても画像は更新されません。



そこで、money.appcacheの「#ver 1.06」を「#ver 1.07」に変更します。その後、「キャッシュ更新チェック」ボタンをチェックします。マニフェストファイルが変更された場合は、次のように更新準備が完了となります。



更新が確認されたら、「キャッシュを更新する」ボタンをクリックします。クリックすると、3秒後にキャッシュが新しいものに更新され、ページ上の画像も新しいものに更新されます。



なお、古いiPhone（iOS 4）では一度キャッシュした内容が削除されず、マニフェストファイルの更新が反映されないことがあります。この場合、iPhoneそのものを完全に初期化すれば、キャッシュを削除することができます。

ONEPOINT

アプリケーションキャッシュを更新するには「swapCache()」メソッドを使う

アプリケーションキャッシュを更新するには、「applicationCache」オブジェクトの「swapCache()」メソッドを使います。ただし、キャッシュデータがダウンロードされていないと更新されません。キャッシュが更新されているか調べて必要なデータをダウンロードするには、「applicationCache」オブジェクトの「update()」メソッドを使います。「update()」メソッドを実行すると、次の表のような各種イベントが発生します。

イベント	説明
checking	チェック中
error	エラー
noupdate	マニフェストファイル更新なし
downloading	データダウンロード中
progress	更新中
cached	キャッシュ完了
obsolete	マニフェストファイルが不明(キャッシュは削除)

キャッシュデータがダウンロードされた場合は、「updateready」イベントが発生します。このイベントが発生したら、「swapCache()」メソッドを使ってデータを更新します。また、更新可能な場合は、「applicationCache」オブジェクトの「status」プロパティの値が「applicationCache.UPDATEREADY」になります。「status」プロパティをチェックした後で更新することもできます。 なお、「status」プロパティの値は、次の表のようになります。

値	キーワード値	説明
0	UNCACHED	キャッシュされていない
1	IDLE	更新なし
2	CHECKING	チェック中
3	DOWNLOADING	ダウンロード中
4	UPDATEREADY	更新完了
5	OBSOLETE	マニフェストファイルが不明のためキャッシュ削除

SECTION-323

タッチ数を求める

ここでは、スマートフォンやタブレットで同時にデバイスにタッチされた数を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

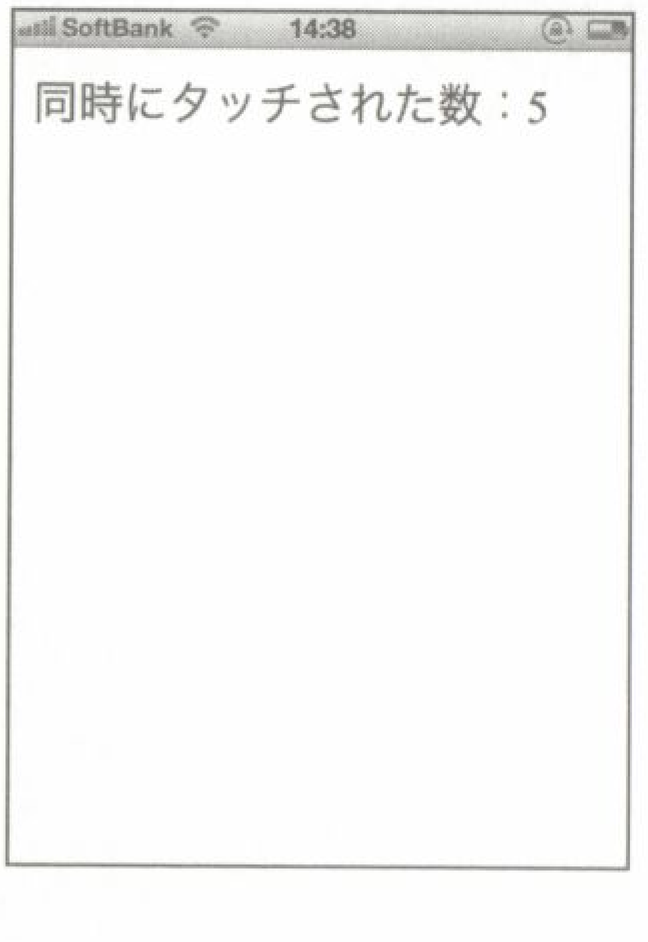
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("touchstart", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  // タッチされた数を求める
  var n = evt.touches.length;
  // 結果を画面に出力
  ele.innerHTML = "同時にタッチされた数 :"+n;
}, false);
```

HINT

この機能は、パソコンではなく、スマートフォンやタブレットが対象です。

ブラウザでindex.htmlを表示し、画面にタッチすると、タッチした数が表示されます。同時タッチできないデバイスもあり、その場合は常に「1」になります。



ONEPOINT 同時タッチ処理を行うにはタッチ関係のイベントを使う

スマートフォンやタブレットでは、複数の指を使って操作(ジェスチャー)することができます。その際、何本の指がタッチされているかを知りたい場合があります。タッチ処理は、次の表に示すイベントが発生します。イベントが発生するとイベントハンドラが呼び出され、イベントオブジェクトが渡されます。このイベントオブジェクトの「touches」配列にタッチされた指の情報が入ります。「touches」配列なので、「length」プロパティを参照すれば同時にタッチされた数を知ることができます。

イベント	説明
touchstart	タッチされた(指が画面に触れた)
touchmove	タッチしたまま指を動かした
touchend	タッチし終わった(指が離された)

COLUMN 同時タッチ数の制限

iOS(iPhone/iPad)では、同時にタッチできる数は5つまでです。Androidの場合は、Android OS 2.xでは1つ、3.x以降は2つ以上、同時にタッチできますが、実際にはいくつまで同時タッチできるかは端末によって異なる場合があります。また、同時タッチに見せかけたような処理が行われることがあるため、Androidでは、同時タッチした際に連続的に「touchstart」イベントが発生するような場合もあります。1カ所のタッチ処理であれば、iOS/Androidなどを問わずに処理することができます。

Windows Phone(7.5)の場合は「touch」イベントが発生しないため、タッチ処理を行うことはできません。また、SafariやGoogle Chromeで使用されているレンダリングエンジンであるWebKitを利用した端末(Kobo Touchなど)でも、タッチイベントは発生しません。

関連項目 ▶▶▶

● タッチされた座標を求める …………………………………………………… p.766

SECTION-324

タッチされた座標を求める

ここでは、スマートフォンやタブレットでタッチした座標値を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

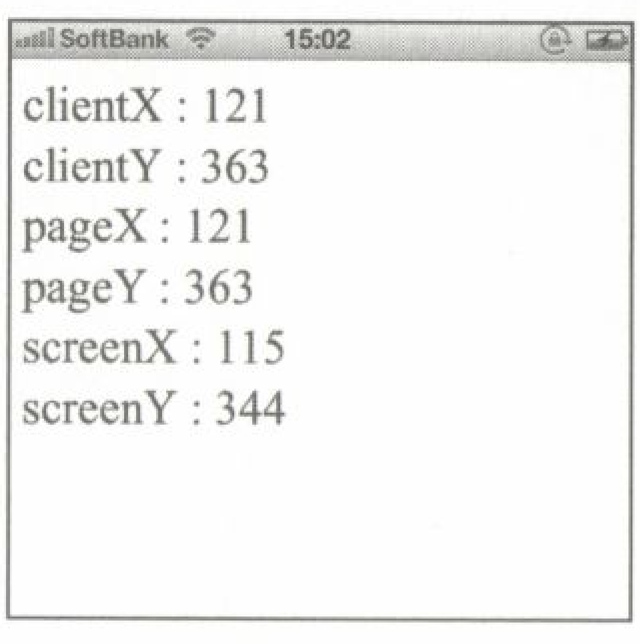
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("touchstart", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  // 最初にタッチされた位置だけ座標値を表示する
  // クライアント要素上でのXY座標
  ele.innerHTML = "clientX : " + evt.touches[0].clientX;
  ele.innerHTML += "<br>clientY : " + evt.touches[0].clientY;
  // ページ上でのXY座標
  ele.innerHTML += "<br>pageX : " + evt.touches[0].pageX;
  ele.innerHTML += "<br>pageY : " + evt.touches[0].pageY;
  // 画面(スクリーン)上でのXY座標
  ele.innerHTML += "<br>screenX : " + evt.touches[0].screenX;
  ele.innerHTML += "<br>screenY : " + evt.touches[0].screenY;
}, false);
```

HINT

この機能は、パソコンではなく、スマートフォンやタブレットが対象です。

ブラウザでindex.htmlを表示し、画面にタッチすると、タッチした座標が表示されます。



ONEPOINT タッチされた座標を調べるには「clientX」「clientY」「pageX」「pageY」「screenX」「screenY」を使う

スマートフォンやタブレットでタッチされた座標値を求めるには、「touches」配列に格納されているオブジェクトの各プロパティを読み出します。読み出せるプロパティは、次の表のようになります。座標値としては、「clientX」「clientY」「pageX」「pageY」「screenX」「screenY」の6つのプロパティがあります。

プロパティ	説明
indetifier	ID(iOSとAndroidではIDの処理が異なる)
target	タッチされたターゲット要素
clientX	クライアントX座標
clientY	クライアントY座標
pageX	ページ上のX座標
pageY	ページ上のY座標
screenX	画面上のX座標
screenY	画面上のY座標

関連項目 ▶▶▶

● タッチ数を求める p.764

SECTION-325

加速度センサーの値を求める

ここでは、スマートフォンやタブレットで加速度センサーの値を求める方法について解説します。

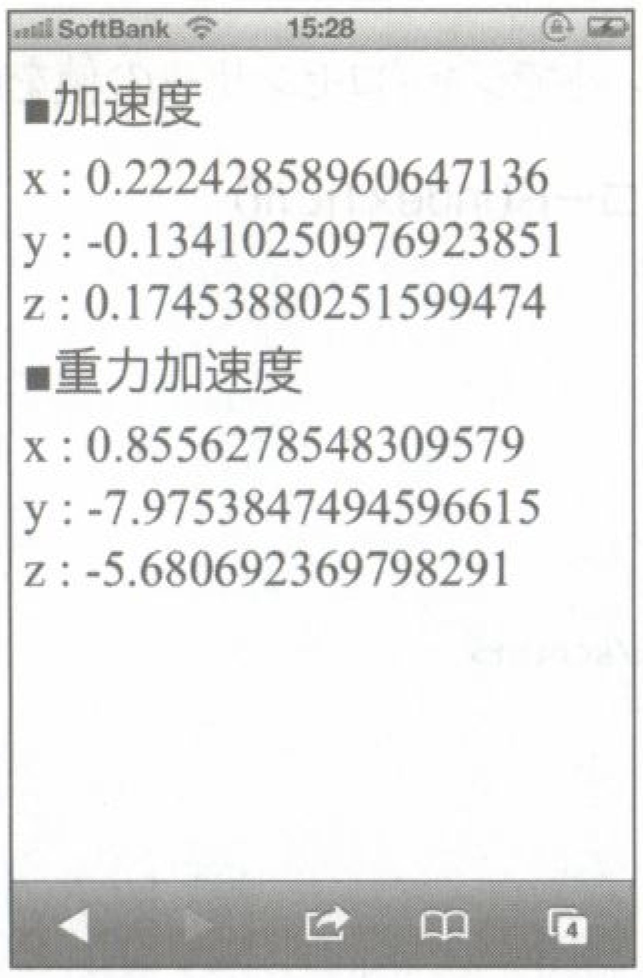
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("devicemotion", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  // 加速度センサーの値
  var x = evt.acceleration.x;
  var y = evt.acceleration.y;
  var z = evt.acceleration.z;
  // 重力加速度センサーの値(iPad/iPhone 4以降)
  var xg = evt.accelerationIncludingGravity.x;
  var yg = evt.accelerationIncludingGravity.y;
  var zg = evt.accelerationIncludingGravity.z;
  // 結果を画面に出力
  ele.innerHTML = "■加速度<br>";
  ele.innerHTML += "x : "+x+"<br>";
  ele.innerHTML += "y : "+y+"<br>";
  ele.innerHTML += "z : "+z+"<br>";
  ele.innerHTML += "■重力加速度<br>";
  ele.innerHTML += "x : "+xg+"<br>";
  ele.innerHTML += "y : "+yg+"<br>";
  ele.innerHTML += "z : "+zg+"<br>";
}, false);
```

ブラウザでindex.htmlを表示し、デバイスを傾けると、それぞれの加速度の値が表示されます。なお、iOS4でもiPad、および、iPhone 4以降でないと重力加速度に関しては動作しません。



ONEPOINT 加速度センサーを扱うには「devicemotion」イベントを使う

スマートフォンやタブレットには、加速度センサーや傾きセンサーなどがあります。加速度センサーの値を調べるには、「devicemotion」イベントを利用します。「devicemotion」イベントは、加速度センサーの値が変化すると発生します。「devicemotion」イベントが発生した際に呼び出すイベントハンドラには、イベントオブジェクトが渡されます。このイベントオブジェクトの「acceleration」オブジェクトに、加速度の値が入っています。加速度の値はそれぞれ方向に応じて、「x」プロパティ、「y」プロパティ、「z」プロパティに格納されています。

プロパティ	説明
acceleration.x	X方向
acceleration.y	Y方向
acceleration.z	Z方向

また、重力加速度の場合は、「accelerationIncludingGravity」オブジェクトの「x」プロパティ、「y」プロパティ、「z」プロパティに格納されています。

プロパティ	説明
accelerationIncludingGravity.x	X方向
accelerationIncludingGravity.y	Y方向
accelerationIncludingGravity.z	Z方向

関連項目 ▶▶▶

●ジャイロセンサーの値を求める ……………………………………………… p.770

SECTION-326

ジャイロセンサーの値を求める

ここでは、スマートフォンやタブレットでジャイロセンサーの値を求める方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("deviceorientation", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  // ジャイロセンサーの値
  var a = evt.alpha;
  var b = evt.beta;
  var g = evt.gamma;
  // 結果を画面に出力
  ele.innerHTML = "■ジャイロセンサー<br>";
  ele.innerHTML += "alpha : "+a+"<br>";
  ele.innerHTML += "beta : "+b+"<br>";
  ele.innerHTML += "gamma : "+g+"<br>";
}, false);
```

ブラウザでindex.htmlを表示し、デバイスを傾けると、それぞれの方向の傾きが角度で表示されます。

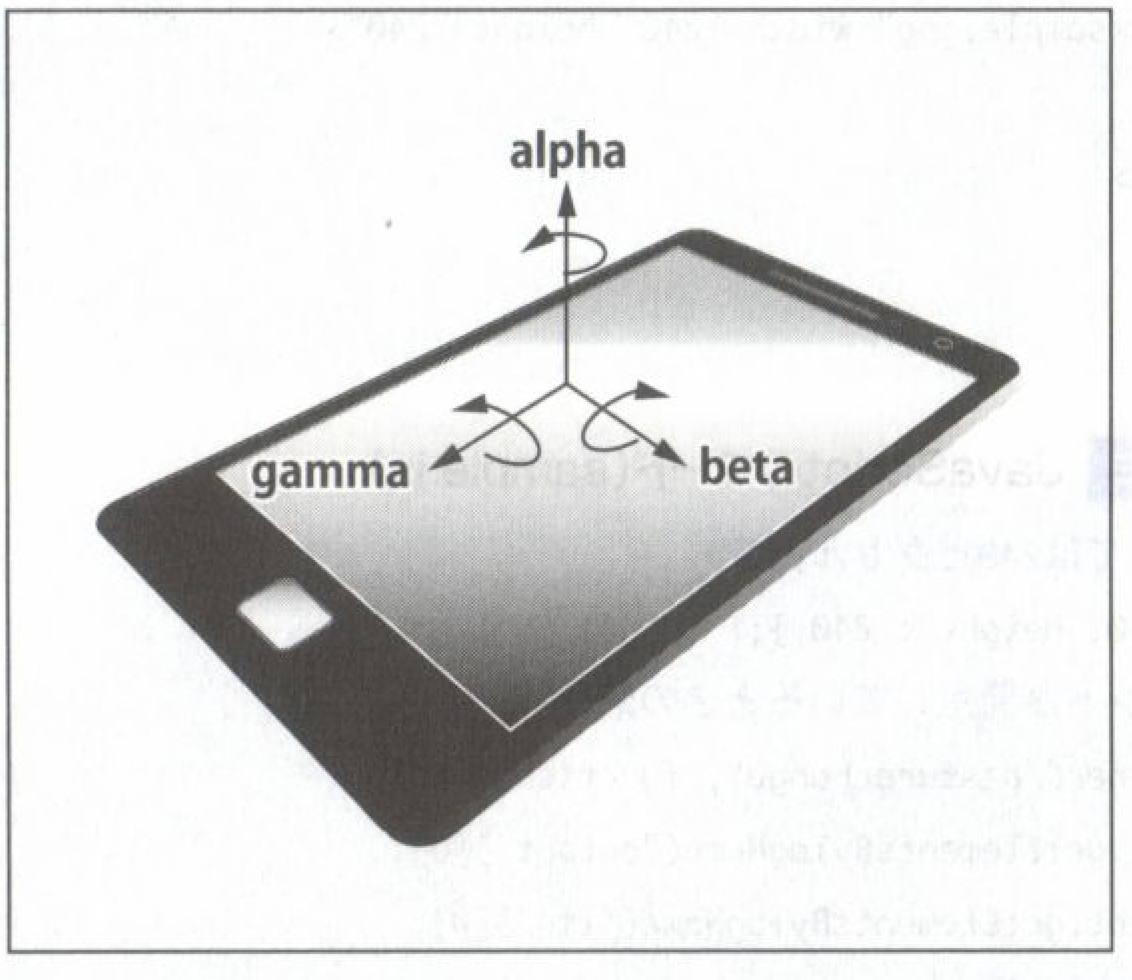


ONEPOINT ジャイロセンサーを扱うには「deviceorientation」イベントを使う

スマートフォンやタブレットには、ジャイロセンサーが装備されている機種があります。ジャイロセンサーの値を調べるには、「deviceorientation」イベントを利用します。「deviceorientation」イベントは、ジャイロセンサーの値が変化すると発生します。「deviceorientation」イベントが発生した際に呼び出すイベントハンドラには、イベントオブジェクトが渡されます。このイベントオブジェクトの「alpha」プロパティ、「beta」プロパティ、「gamma」プロパティに、次の表のように、それぞれの軸の傾きが入っています。値は傾きの角度(ラジアンではない)が入っています。

プロパティ	説明	値の範囲
alpha	画面奥から手前方向の軸	0〜360
beta	横軸	-90〜90
gamma	縦軸	-90〜270

図で表すと、次のようになります。



関連項目 ▶▶▶

● 加速度センサーの値を求める……p.768

SECTION-327

ジェスチャーされたら拡大や回転処理を行う

ここでは、iOS(iPhone/iPad)でジェスチャーされたら拡大や回転処理を行う方法について解説します。

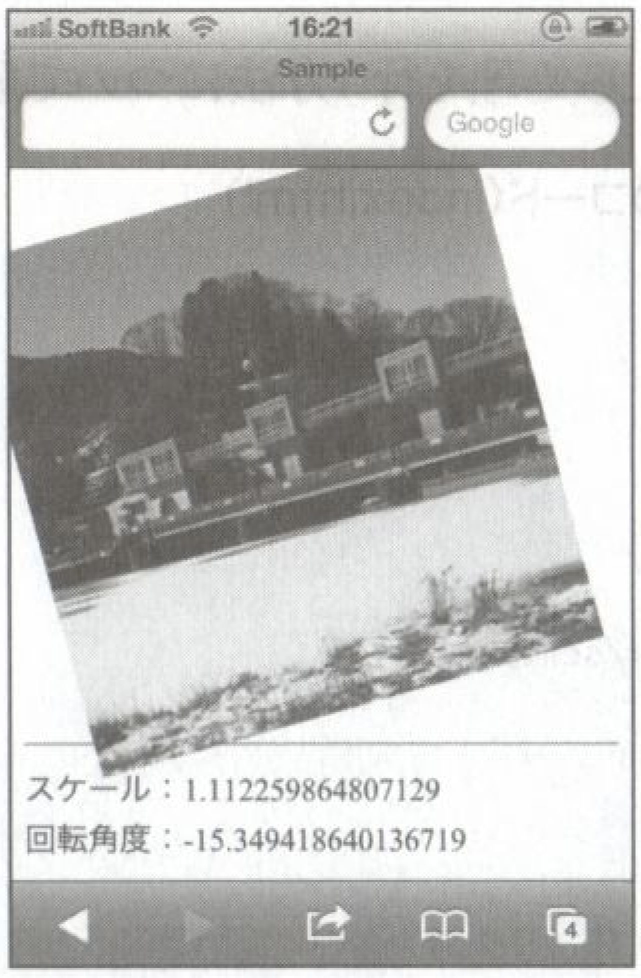
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content= "initial-scale=1, user-scalable=no">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <img src="images/sample.jpg" width="240" height="240">
    <hr>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
// 画像のサイズ(ここでは240ピクセル固定)
var g = { width : 240, height : 240 };
// ジェスチャーイベントが発生しているときの処理
window.addEventListener("gesturechange", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  var photo = document.getElementsByTagName("img")[0];
  // スケール(拡大縮小)の値
  var s = evt.scale;
  // 回転角度
  var r = evt.rotation;
  // 画像を拡大縮小
  photo.style.width = g.width * s + "px";
  photo.style.height = g.height * s + "px";
  // 画像を回転
  photo.style.webkitTransform = "rotate(" + r + "deg)";
  // 結果を画面に出力
  ele.innerHTML = "スケール :"+s;
  ele.innerHTML += "<br>回転角度 :"+r;
}, false);
```

ブラウザでindex.htmlを表示し、画面上でジェスチャー（2本指で操作）すると、画像がジェスチャー状況に合わせて回転したりサイズが変わります。また、どのくらい回転したか、拡大・縮小したかが数値で表示されます。



ONEPOINT

ジェスチャーによる回転と拡大・縮小を行うには「gesturechange」イベントなどを使う

iOSでは、複数の指で画面をタッチした場合に、「touchstart」「touchmove」「touchend」とは別に、次の表に示すジェスチャーイベントが発生します。

イベント	説明
gesturestart	開始
gesturechange	ジェスチャー中
gestureend	ジェスチャー終了

ジェスチャーイベントが発生した際に呼び出されるイベントハンドラには、イベントオブジェクトが渡されます。このイベントオブジェクトの「scale」プロパティに、どのくらい指が離されたか、つまり、拡大・縮小率が入ります。また、指で回すような操作をする、つまり、回転させた場合には、回転角度がイベントオブジェクトの「rotation」プロパティに入ります。「rotation」プロパティに入るのは角度の値であり、ラジアンではありません。

画像の元の横幅と縦幅に「scale」プロパティの値を乗算し、スタイルシートの「width」プロパティ、「height」プロパティに設定すれば、ジェスチャーによって画像を拡大・縮小することができます。

同様に、「rotation」プロパティによって得られた回転角度をスタイルシートの「webkit Transform」プロパティに設定すれば、画像を回転させることができます。

関連項目 ▶▶▶

● イベントハンドラを設定する ………………………………… p.396
● イベントリスナーを設定する ………………………………… p.398

SECTION-328

デバイスが回転したら処理を行う

ここでは、デバイスが回転したら処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <output></output>
  </body>
</html>
```

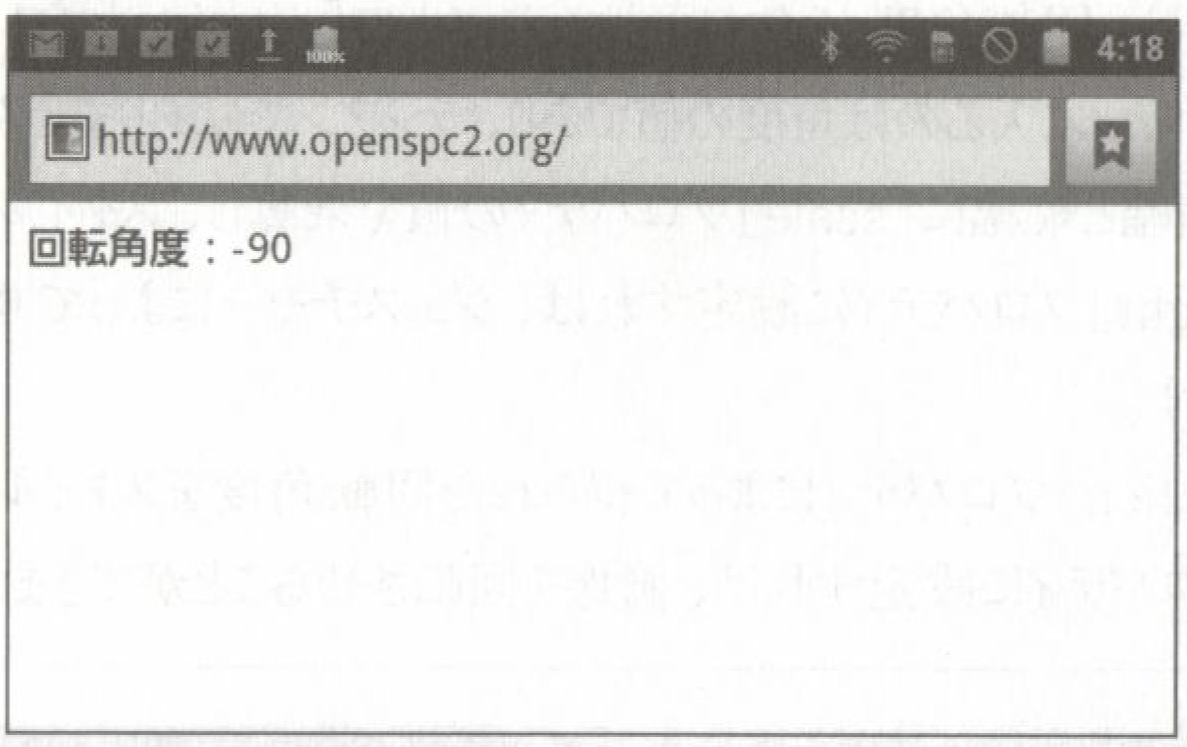
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("orientationchange", function(evt){
  var ele = document.getElementsByTagName("output")[0];
  // 回転角度を求める
  var deg = window.orientation;
  // 結果を画面に出力
  ele.innerHTML = "回転角度：" +deg;
}, false);
```

HINT

この機能は、パソコンではなく、スマートフォンやタブレットが対象です。

ブラウザでindex.htmlを表示し、デバイスを回転すると、回転方向を示す角度が表示されます。



ONEPOINT

デバイスの回転時に処理を行うには「orientationchange」イベントを使う

スマートフォンではデバイスを回転させた際、「orientationchange」イベントが発生します。「orientationchange」イベントが発生した際、デバイスが縦向きか横向きかを知りたい場合は、「window」オブジェクトの「orientation」プロパティの値を参照します。この値は次の表に示す4種類を返します。なお、iPadでは「180」の値が返されますが、iPhoneでは180度回転しても対応していないため、「180」の値が返されることはありません。デバイスの向きを知りたい場合には、ジャイロセンサーの値を読み出す必要があります。

値	説明
0	縦向き
90	右に回転
-90	左に回転
180	180度回転

関連項目 ▶▶▶

●イベントハンドラを設定する ……… p.396
●イベントリスナーを設定する ……… p.398

SECTION-329

日付の入力を行うようにする

ここでは、日付の入力を行うようにする方法について解説します。

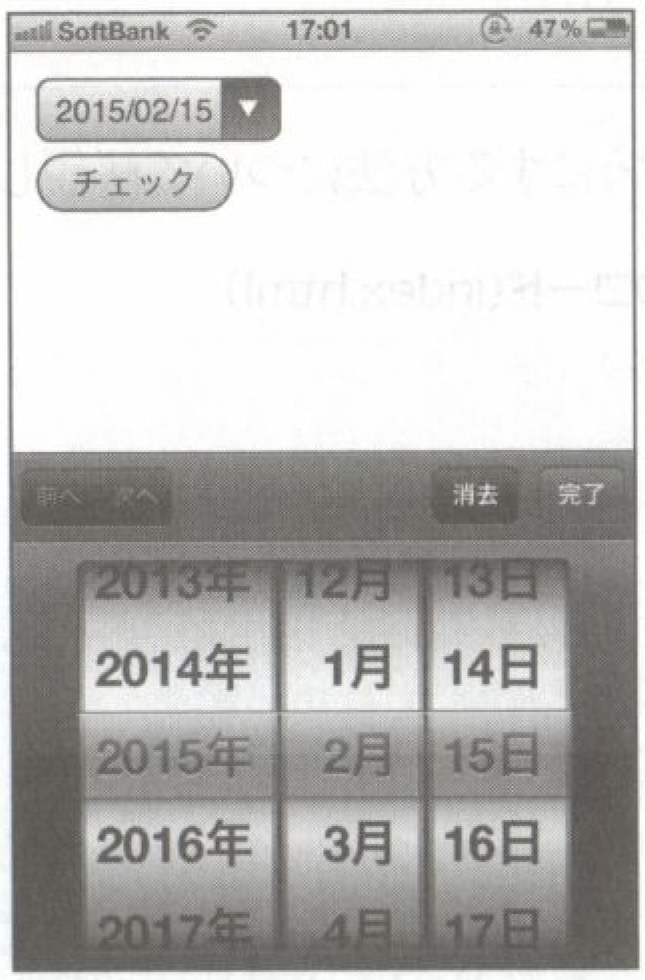
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="date" id="myDate"><br>
      <input type="button" id="check" value="チェック">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタン要素の読み出し
  var btn = document.getElementById("check");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 入力された日付を取得
    var text = document.getElementById("myDate").value;
    var dtext = document.getElementById("myDate").valueAsDate;;
    // 結果を出力
    ele.innerHTML = "value:"+text+"<br>";
    ele.innerHTML += "valueAsDate:"+dtext;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィールドをクリックすると、日付を選択するカレンダーなどが表示されます。



「チェック」ボタンをクリックすると、選択した日付の文字列が表示されます。



ONEPOINT 日付を選択するには「input」要素の「type」属性に「date」を指定する

日付を選択するには、「input」要素の「type」属性に「date」を指定します。「date」を指定したフィールドをクリックすると、カレンダー、または、スピンボタンによる日付選択などが可能になります。どのような選択画面になるかは、ブラウザによって異なります。

選択された日付を読み出すには、「value」プロパティにアクセスします。日付文字列として取得したい場合には、「valueAsDate」プロパティにアクセスします。

関連項目 ▶▶▶

● 内容をDate型/数値型として読み出す …… p.327
● 日時の入力を行うようにする …… p.778

SECTION-330

日時の入力を行うようにする

ここでは、日時の入力を行うようにする方法について解説します。

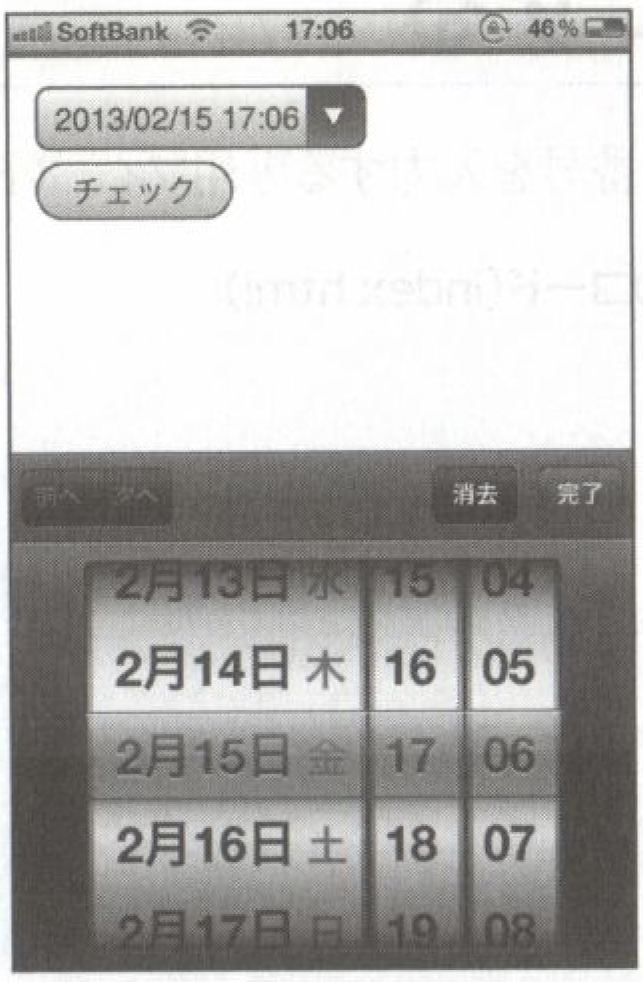
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="datetime" id="myDate"><br>
      <input type="button" id="check" value="チェック">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタン要素の読み出し
  var btn = document.getElementById("check");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 入力された日時を取得
    var text = document.getElementById("myDate").value;
    var dtext = document.getElementById("myDate").valueAsDate;;
    // 結果を出力
    ele.innerHTML = "value:"+text+"<br>";
    ele.innerHTML += "valueAsDate:"+dtext;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィールドをクリックすると、日時を選択するカレンダーなどが表示されます。



「チェック」ボタンをクリックすると、選択した日付と時間の文字列が表示されます。



ONEPOINT **日付と時間をまとめて選択するには「input」要素の「type」属性に「datetime」を指定する**

日付と時間をまとめて選択するには、「input」要素の「type」属性に「datetime」を指定します。「datetime」を指定したフィールドをクリックすると、カレンダー、または、スピンボタンによる日付選択などが可能になります。どのような選択画面になるかは、ブラウザによって異なります。

選択された日付を読み出すには、「value」プロパティにアクセスします。日付文字列として取得したい場合には、「valueAsDate」プロパティにアクセスします。

関連項目 ▶▶▶

- 内容をDate型/数値型として読み出す …… p.327
- 日付の入力を行うようにする …… p.776

SECTION-331

メールやURL、電話番号を入力する専用フィールドにする

ここでは、メールやURL、電話番号を入力する専用フィールドにする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p>メールアドレス：<input type="email" id="adrs"></p>
      <p>   ウェブサイト：<input type="url" id="website"></p>
      <p>連絡先電話番号：<input type="tel" id="telephone"></p>
      <input type="button" id="check" value="確認">
      <input type="submit" id="check" value="送信">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタン要素の読み出し
  var btn = document.getElementById("check");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 入力されたメールアドレスを取得
    var userMail = document.getElementById("adrs").value;
    // 入力されたURLを取得
    var userSite = document.getElementById("website").value;
    // 入力された電話番号を取得
    var userTel = document.getElementById("telephone").value;
    // 結果を出力
    ele.innerHTML = "メール：" +userMail+"<br>";
    ele.innerHTML += "ウェブ：" +userSite+"<br>";
    ele.innerHTML += "電話番号：" +userTel;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィールドに文字を入力して「確認」クリックすると、入力内容が表示されます。



「送信」ボタンをクリックすると、チェックが行われ、正しくない項目には再入力を促すメッセージが表示されます。





ONEPOINT

メールアドレスやURLの入力専用フィールドにするには「input」要素の「type」属性に「email」や「url」を指定する

メールアドレスの入力専用フォールドにするには、「input」要素の「type」属性に「email」を指定します。「email」を指定すると、「xxxx@xxx.xxx」といったメールアドレスとして認識できる形式以外は不正とみなされ、データが送信できなくなります。入力したメールアドレスが不正な場合は、送信時にブラウザがメッセージを表示します（プログラムによってもチェックすることができる）。

同様に、URL入力専用にする場合は、「type」属性に「url」を指定します。電話番号の場合は「type」属性に「tel」を指定します。電話番号の場合は、スマートフォンでは電話番号入力専用のソフトウェアキーボードが表示されます。なお、電話番号形式の整合性はチェックされません。

また、検索キーワードを示すフィールドにする場合は、「type」属性に「search」を指定します。

SECTION-332

数値入力・スライダー入力の専用フィールドにする

ここでは、数値入力・スライダー(範囲指定)入力の専用フィールドにする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <p>注文番号：<input type="number" value="596" id="orderNo"></p>
      <p>注文個数：<input type="range" value="5" min="1" max="10" id="count"></p>
      <input type="button" id="check" value="確認">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタン要素の読み出し
  var btn = document.getElementById("check");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 入力された番号を取得
    var no1 = document.getElementById("orderNo").value;
    var no2 = document.getElementById("orderNo").valueAsNumber;
    // 入力された個数を取得
    var cnt1 = document.getElementById("count").value;
    var cnt2 = document.getElementById("count").valueAsNumber;
    // 結果を出力
    ele.innerHTML = "注文番号(value):"+no1+"<br>";
    ele.innerHTML += "注文番号(valueAsNumber):"+no2+"<br>";
    ele.innerHTML += "注文個数(value):"+cnt1+"<br>";
    ele.innerHTML += "注文個数(valueAsNumber):"+cnt2+"<br>";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、数値入力専用フィールドやスライダーを調整した後、「確認」ボタンをクリックすると、フィールドの値が表示されます。



ONEPOINT

数値入力専用にするには「input」要素の「type」属性で「number」を指定する

フィールドを数値入力専用にするには、「input」要素の「type」属性で「number」を指定します。「number」を指定すると、ブラウザによっては、フィールドの横にスピンボタンが表示されます。スマートフォンでは数値だけ入力可能なソフトウェアキーボードが表示されます。

また、直接、数値を入力するのではなく、スライダーを指定することもできます。スライダーを表示する場合は、「type」属性に「range」を指定します。スライダーの数値範囲を指定する場合、最小値を「min」、最大値を「max」で指定することができます。スライダーに対応していないブラウザでは、従来通りのテキストフィールドが表示されます。

数値入力の場合でも、JavaScriptで「value」プロパティを使って値を読み出すと、文字型になってしまいます。数値型として読み出すには、「valueAsNumber」プロパティから読み出します。

関連項目 ▶▶▶

● 内容をDate型/数値型として読み出す …… p.327
● 数値専用のフィールドで値を増減する …… p.335

SECTION-333

カラーピッカーを表示する

ここでは、カラーピッカーを表示する方法について解説します。

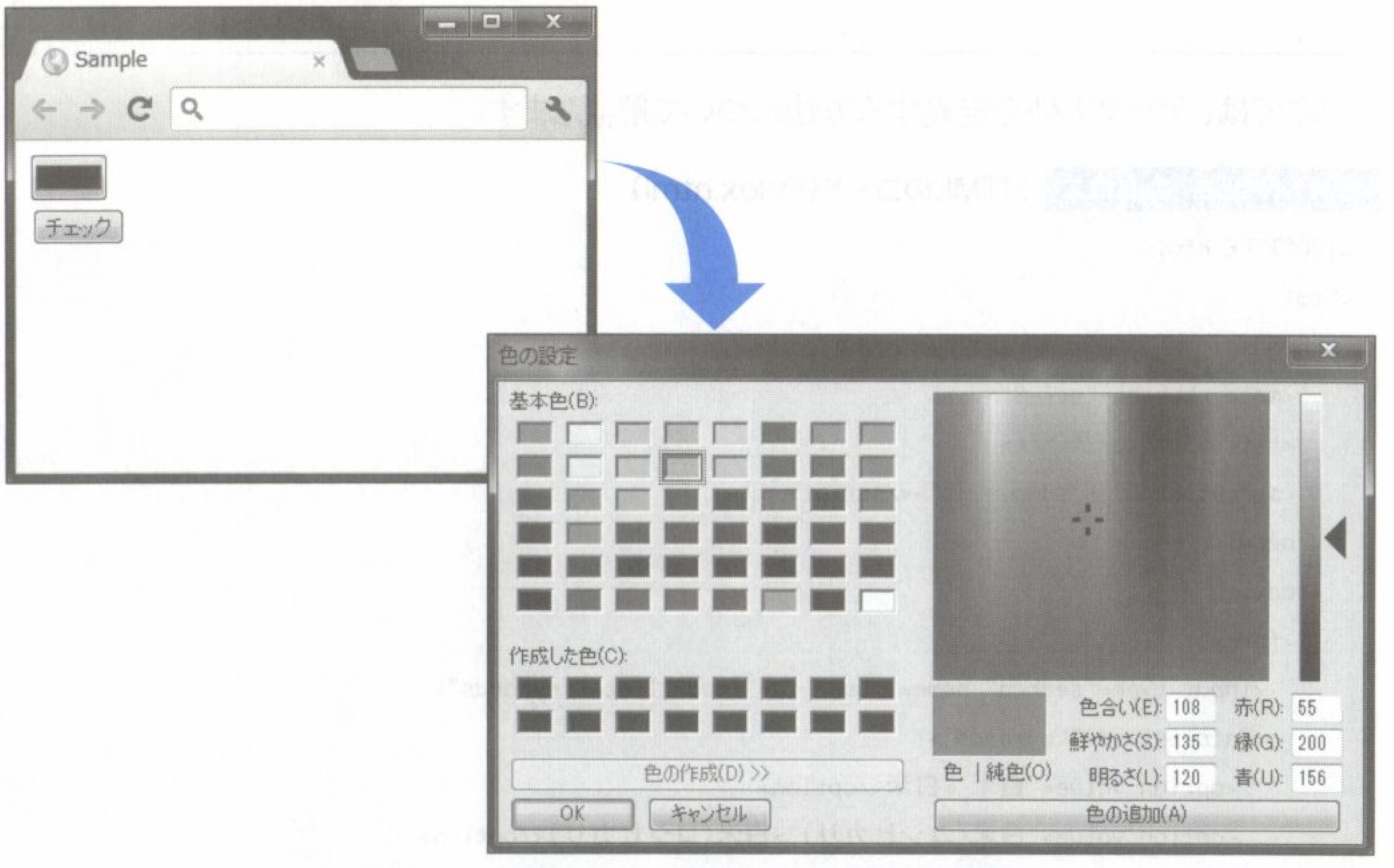
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="color" id="myColor"><br>
      <input type="button" id="check" value="チェック">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタン要素の読み出し
  var btn = document.getElementById("check");
  // ボタンにイベントを割り当て
  btn.addEventListener("click", function(evt){
    // 選択されたカラーコードを取得
    var color = document.getElementById("myColor").value;
    // 結果を出力
    ele.innerHTML = "カラーコード :"+color;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、「type="color"」を指定した要素はセレクトメニューと似たような形状になり、クリックするとカラーを選択できます。



「チェック」ボタンをクリックすると、選択した色のカラーコードが表示されます。



ONEPOINT

カラーピッカーを表示するには「input」要素の「type」属性に「color」を指定する

カラーピッカーを表示するには、「input」要素の「type」属性に「color」を指定します。「color」を指定した場合、すべてのカラーを選択できるわけではありません。あらかじめ、用意されているカラーから選択することになります。その際、「datalist」要素と組み合わせることで、任意のカラーを表示することができます。

SECTION-334

データリストを定義する

ここでは、データリストを定義する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="search" name="food" id="food" list="keywords">
      <datalist id="keywords">
        <option value="白米">白米</option>
        <option value="白米(コシヒカリ)">白米(コシヒカリ)</option>
        <option value="白米(とどろき早生)">白米(とどろき早生)</option>
        <option value="玄米">玄米</option>
        <option value="五穀米">五穀米</option>
      </datalist>
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // フィールドの読み出し
  var fld = document.getElementById("food");
  // フィールドにイベントを割り当て
  fld.addEventListener("change", function(evt){
    // 選択された内容を読み出し
    var text = this.value;
    // 結果を出力
    ele.innerHTML = "選択した項目 :"+text;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールドに文字を入力すると、該当するリストが表示されます。



候補から選択すると、フィールド内とフィールドの下に選択した文字が表示されます。



ONEPOINT データリストを表示するには「datalist」要素を使う

フィールドに文字を入力する際に、あらかじめ入力される可能性が高いものをデータリストとして用意しておくことができます(オートコンプリート/サジェスト機能)。データリストは特別なスクリプトを記述する必要はなく、HTMLの「datalist」要素を使って定義します。「datalist」要素内に「option」要素を使い、表示される項目を列記します。また、テキストフィールドでは「list」属性に「datalist」要素のID名を指定することで、対応するデータリストを指定することができます。データリストを利用して入力されたテキストフィールドの内容は、「value」プロパティを使って読み出すことができます。

SECTION-335

音声入力を行う

ここでは、音声入力を行う方法と、入力完了後に処理する方法について解説します。

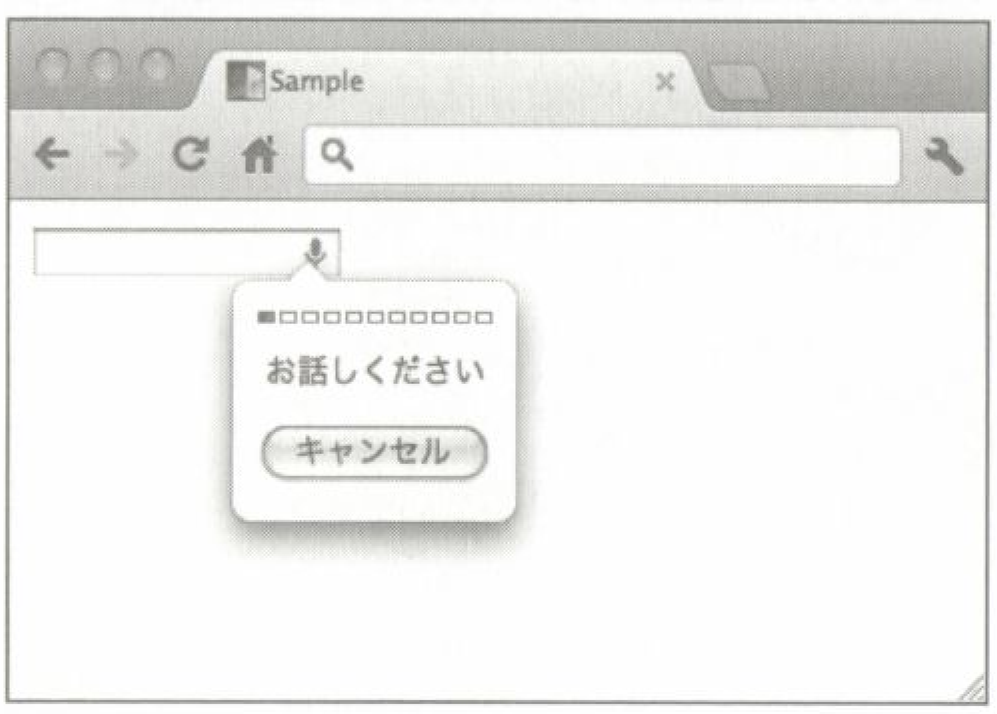
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <form>
      <input type="text" id="myVoice" x-webkit-speech>
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // input要素の読み出し
  var voice = document.getElementById("myVoice");
  // イベントを割り当て
  voice.addEventListener("webkitspeechchange", function(evt){
    // 音声入力された文字を出力
    ele.innerHTML = voice.value;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、テキストフィールド右側にあるマイクアイコン(ボタン)をクリックすると、マイクのふきだしが表示されるので、マイクに向かって話します。



音声の入力が完了すると、イベントが発生し、音声入力された文字がテキストフィールドの下に表示されます。



ONEPOINT

音声入力を行うには「input」要素に「x-webkit-speech」属性を指定する

音声入力を行うには、「input」要素に「x-webkit-speech」属性を指定します。これで自動的にテキストフィールドの右側にマイクアイコン(ボタン)が表示されます。このマイクアイコンをクリックすると、音声入力を行うための吹き出しが表示されます。吹き出しが表示されてから話した内容が解析され、テキストフィールドに入ります。なお、音声入力後の解析処理はサーバー側で行われるので、ネットワークに接続されていることが条件になります。

また、音声入力が完了した場合は、「webkitspeechchange」イベントが発生します。このイベントが発生したら、テキストフィールドの内容を読み出して処理を行います。

SECTION-336

ブラウザのページ読み込みタイミングを取得する

ここでは、ブラウザのページ読み込みタイミングを取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>タイミング表示</h1>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // 読み込み種類で表示する文字を配列に設定
  var loadType = ["クリックまたはURL入力", "リロード", "進む／戻るボタン"];
  // 読み込み種類を取得
  var nType = performance.navigation.type;
  ele.innerHTML = "種類 :"+loadType[nType]+"<hr>";
  // 各種タイミングに関するプロパティを配列に設定
  var param = [
    "navigationStart", "unloadEventStart", "unloadEventEnd", "redirectStart",
    "redirectEnd", "fetchStart", "domainLookupStart", "domainLookupEnd",
    "connectStart", "connectEnd", "secureConnectionStart", "requestStart",
    "responseStart", "responseEnd", "domLoading", "domInteractive",
    "domContentLoadedEventStart", "domContentLoadedEventEnd",
    "domComplete", "loadEventStart", "loadEventEnd"
  ];
  // 現在の時間と開始時間を取得
  var now = (new Date()).getTime();
  var startTime = performance.timing.navigationStart;
  // それぞれのタイミング値を表示
  for(var i=0; i<param.length; i++){
    // タイミングプロパティを読み出し
    var measureTime = performance.timing[param[i]];
    if (measureTime - startTime >= 0){
      ele.innerHTML += param[i]+" : "+(measureTime - startTime)+"ミリ秒<br>";
```

```
    }else{
      // 取得できない場合の処理
      ele.innerHTML += param[i]+" : 不明<br>";
    }
  }
}, false);
```

ブラウザでindex.htmlを表示し、ページが読み込まれると、どのような方法で読み込まれたか、どのようなタイミングで各種情報が読み込まれたかが表示されます。



ONEPOINT

読み込みタイミングを取得するには「performance.navigation」オブジェクトの各種プロパティを使う

読み込みタイミングを取得するには、「performance.navigation」オブジェクトの各種プロパティを参照します。プロパティには、その処理が行われた時点での時間を示すミリ秒が入ります。このミリ秒は、実行した時点での日時を示します。

状況によっては、タイミングを正しく取得できない場合があります。この場合、ページ読み込み開始時間よりも以前の時間（ミリ秒）になります。

プロパティ	説明
navigationStart	ページ処理開始
unloadEventStart	ページアンロードイベント開始
unloadEventEnd	ページアンロードイベント終了
redirectStart	リダイレクト開始
redirectEnd	リダイレクト終了
fetchStart	フェッチ開始
domainLookupStart	DNSへの問い合わせ開始
domainLookupEnd	DNSへの問い合わせ終了
connectStart	接続開始
connectEnd	接続終了
secureConnectionStart	セキュア接続開始 （セキュアサーバー利用の場合のみ）
requestStart	サーバーへのリクエスト開始
responseStart	レスポンス開始
responseEnd	レスポンス終了
domLoading	DOM読み込み
domInteractive	DOM処理
domContentLoadedEventStart	DOMContentReadyイベント開始
domContentLoadedEventEnd	DOMContentReadyイベント終了
domComplete	DOM構築完了
loadEventStart	ページloadイベント開始
loadEventEnd	ページloadイベント終了

また、どのような方法（URL指定かリロードかなど）で読み込まれたかは、「performance.navigation」オブジェクトの「type」プロパティで取得することができます。

値	キーワード値	説明
0	TYPE_NVIGATE	クリックされたかURLを入力してページに移動
1	TYPE_RELOAD	リロードした
2	TYPE_BACK_FORWARD	ブラウザの「進む」「戻る」ボタンでページ移動

COLUMN ミリ秒以下の計測

ミリ秒以下の計測を行うための仕様の策定も行われています。計測時間は、「DOMHighResTimeStamp」型として定義され、「now()」メソッドを使って時間を取得することができます。

URL http://www.w3.org/TR/2012/CR-hr-time-20120522/

SECTION-337

異なるオリジンのインラインフレームにメッセージを送信する

ここでは、クロスドキュメントメッセージングを使って異なるオリジンのインラインフレームにメッセージを送信する方法について解説します。

SAMPLE CODE HTMLのコード/送信側(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>クロスドキュメントメッセージ</h1>
    <div>iframeに1秒ごとに日付情報を送信</div>
    <iframe id="clock" src="../sample_iframe/" width="360" height="100">
    </iframe>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード/送信側(sample.js)

```
window.addEventListener("load", sendDateTime, false);
function sendDateTime(){
  // メッセージを送信するiframe要素
  var targetFrame = document.getElementById("clock");
  // 現在時刻を取得
  var dateTime = (new Date()).toString();
  // 送信先のオリジンを設定。*ならオリジン問わない
  var targetOrigin = "*";
  // iframe要素に現在時刻を送信
  targetFrame.contentWindow.postMessage(dateTime, targetOrigin);
  // 1秒ごとに繰り返し送信
  setTimeout(arguments.callee, 1000);
}
```

SAMPLE CODE HTMLのコード/受信側(index.html)

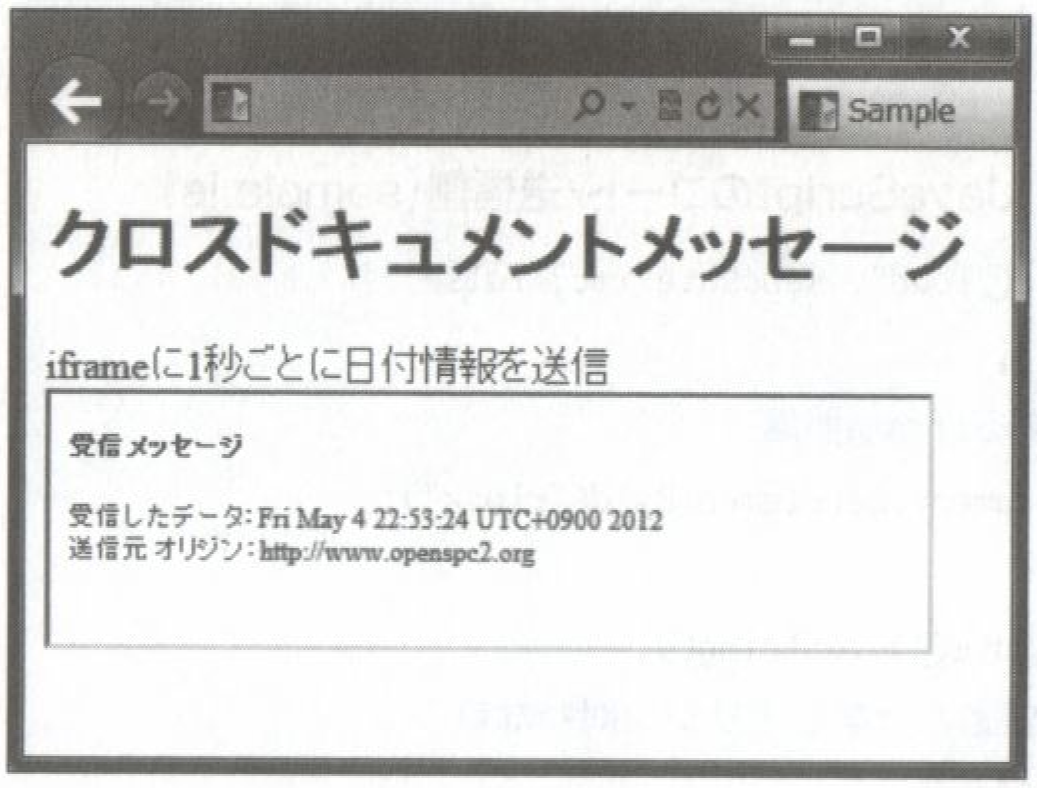
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>body { font-size: 70%; }</style>
    <script src="js/sample.js"></script>
```

```
    </head>
    <body>
      <h4>受信メッセージ</h4>
      <div>受信したデータ：<output></output></div>
      <div>送信元オリジン：<output></output></div>
    </body>
</html>
```

SAMPLE CODE JavaScriptのコード/送信側(sample.js)

```
window.addEventListener("message", function(evt){
  var ele = document.getElementsByTagName("output");
  // 受信したメッセージを画面に出力
  ele[0].innerText = ele[0].textContent = evt.data;
  // 送信元のオリジンを画面に出力
  ele[1].innerText = ele[1].textContent = evt.origin;
}, false);
```

ブラウザで送信側のindex.htmlを表示すると、異なるオリジンにあるインラインフレームにメッセージ(データは現在の日時)を送信します。受信側では、送信されたデータをページ上に表示します(ただし、受信側のindex.htmlを表示しても何も表示されない)。



ONEPOINT インラインフレームにメッセージを送信するには「postMessage()」メソッドを使う

インラインフレームにメッセージを送信するには、「postMessage()」メソッドを使います。「postMessage()」メソッドを使うと、異なるオリジン(プロトコル+ドメイン+ポート番号)に対してデータを送信することができます。「postMessage()」メソッドの最初のパラメータには、送信するデータを指定します。2番目のパラメータには、送信先のオリジンを文字列で指定します。特に指定する必要がない場合には、「"*"」とします。

「postMessage()」メソッドでデータを送信した場合、受信側のインラインフレームでは「message」イベントが発生します。「onmessage」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントリスナーを設定します。

イベントハンドラには、イベントオブジェクトが渡されます。イベントオブジェクトの「data」プロパティに、受信したデータが入っています。また、受信したデータがどこのオリジンのものか確認する場合には、イベントオブジェクトの「origin」プロパティを参照します。

SECTION-338

映像をフルスクリーン表示にする

ここでは、ページ内に表示されている映像をフルスクリーン表示にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>フルスクリーン表示</h3>
    <video width="384" height="216" controls>
      <source src="video/waterfall.mov">
      <source src="video/waterfall.ogv">
    </video>
    <form>
      <input type="button" value="全画面にする" id="setFS">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setFS").addEventListener("click", function(){
    var myVideo = document.getElementsByTagName("video")[0];
    // Safari/Google Chrome用
    if(myVideo.webkitRequestFullScreen){
      myVideo.webkitRequestFullScreen();
    }
    // Firefox用
    if (myVideo.mozRequestFullScreen){
      myVideo.mozRequestFullScreen();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「全画面にする」ボタンをクリックすると、映像がフルスクリーン（全画面）で再生されます。



フルスクリーン表示の場合、最初に画面上に元に戻す方法が表示されます。escキーを押すと元の状態に戻ります。



ONEPOINT 映像をフルスクリーンにするには「video」要素の「requestFullscreen()」メソッドを使う

映像をフルスクリーンにするには、「video」要素の「requestFullscreen()」メソッドを使います。フルスクリーン表示が可能なブラウザであれば、映像がフルスクリーンで再生されます。また、フルスクリーン再生開始時には元の画面に戻す方法が画面に一時的に表示されます。

仕様では、フルスクリーン状態を元に戻すには、「exitFullscreen()」メソッドを使います。

COLUMN Fullscreen APIのベンダープレフィックス

2012年現在、Fullscreen APIはベンダープレフィックス付きで使用することができます。また、仕様ではFullScreenのSが小文字になっていますが、ベンダープレフィックス付きのものは大文字になっています。SafariやGoogle ChromeなどのWebKit系では「webkitRequestFullScreen()」メソッド、Firefoxでは「mozRequestFullScreen()」メソッドを使ってフルスクリーンモードにすることができます。

フルスクリーンモードを解除するには、SafariやGoogle Chromeでは「webkitCancelFullScreen()」メソッド、Firefoxでは「mozCancelFullScreen()」メソッドを使います。

関連項目 ▶▶▶

- 画像をフルスクリーン表示にする …… p.799
- テキストをフルスクリーン表示にする …… p.802
- インラインフレームをフルスクリーン表示にする …… p.805
- フルスクリーンと通常表示が切り替わったら処理を行う …… p.808

SECTION-339

画像をフルスクリーン表示にする

ここでは、ページ内に表示されている画像をフルスクリーン表示にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

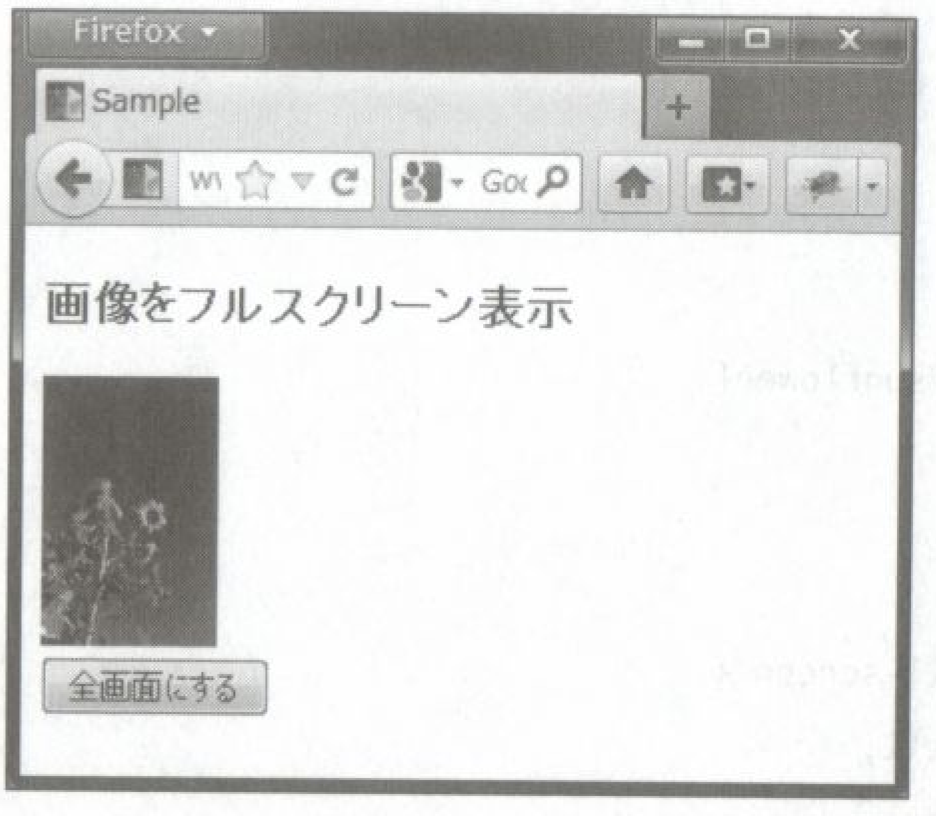
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      #sunflower:-webkit-full-screen {
        width: 720pt;
        height: 1080pt;
      }
      :-moz-full-screen #sunflower{
        width: 720pt;
        height: 1080pt;
      }
      #container:-moz-full-screen {
        display: table-cell;
        text-align: center;
        vertical-align: middle;
      }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>画像をフルスクリーン表示</h3>
    <div id="container">
    <img src="images/photo.jpg" id="sunflower" alt="ひまわり" width="72" height="108">
    </div>
    <form>
      <input type="button" value="全画面にする" id="setFS">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setFS").addEventListener("click", function(){
    var myImage = document.getElementById("sunflower");
```

```
      // Safari/Google Chrome用
      if(myImage.webkitRequestFullScreen){
        myImage.webkitRequestFullScreen();
      }
      // Firefox用
      if (myImage.mozRequestFullScreen){
        myImage.parentNode.mozRequestFullScreen();
      }
    }, false);
  }, false);
```

ブラウザでindex.htmlを表示し、「全画面にする」ボタンをクリックすると、画像がフルスクリーン（全画面）になります。



フルスクリーン表示の場合、最初に画面上に元に戻す方法が表示されます。escキーを押すと元の状態に戻ります。



ONEPOINT

画像をフルスクリーンにするには「img」要素の「requestFullscreen()」メソッドを使う

画像をフルスクリーンにするには、「img」要素の「requestFullscreen()」メソッドを使います。ただし、SafariやGoogle ChromeとFirefoxの両方で同じように動作させるには、スタイルシート、および、HTML要素に工夫が必要です。まず、Firefoxに対応させる場合には、「img」要素を囲む親要素(「div」要素など)が必要になります。SafariやGoogle Chromeの場合は直接、「img」要素の「requestFullscreen()」メソッドを呼び出せばフルスクリーンになりますが、Firefoxではdiv要素などで囲み、さらにスタイルシートを設定する必要があります。Firefoxの場合は「img」要素と、親要素の両方にフルスクリーンモード時の表示位置やサイズなどを設定します。

COLUMN

Fullscreen APIのベンダープレフィックス

2012年現在、Fullscreen APIはベンダープレフィックス付きで使用することができます。また、仕様ではFullScreenのSが小文字になっていますが、ベンダープレフィックス付きのものは大文字になっています。SafariやGoogle ChromeなどのWebKit系では「webkitRequestFullScreen()」メソッド、Firefoxでは「mozRequestFullScreen()」メソッドを使ってフルスクリーンモードにすることができます。

関連項目 ▶▶▶

- 映像をフルスクリーン表示にする …… p.796
- テキストをフルスクリーン表示にする …… p.802
- インラインフレームをフルスクリーン表示にする …… p.805
- フルスクリーンと通常表示が切り替わったら処理を行う …… p.808

SECTION-340

テキストをフルスクリーン表示にする

ここでは、ページ内に表示されているテキストなどの領域(セクションなど)をフルスクリーン表示にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

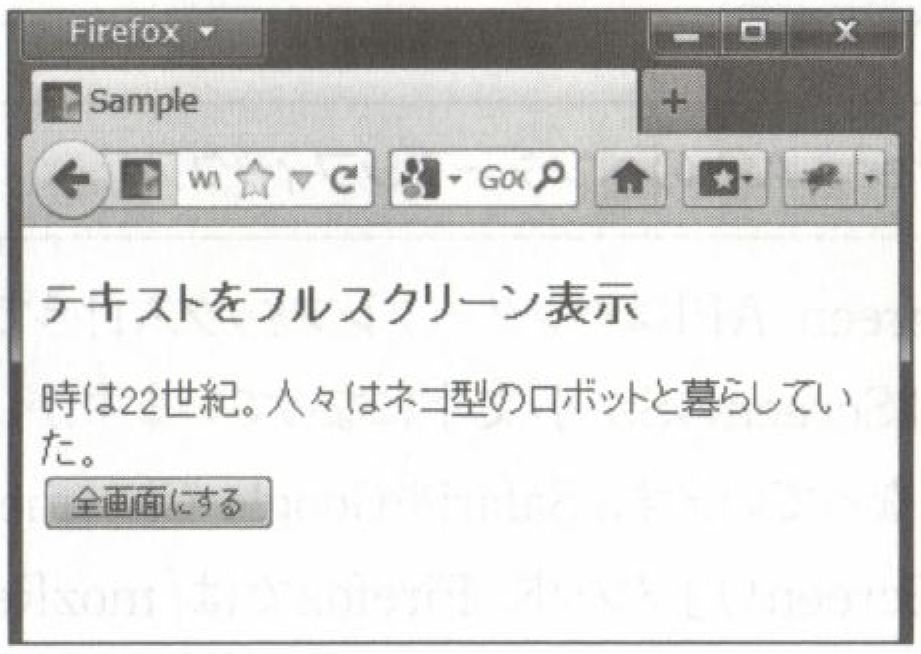
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      #mainText:-webkit-full-screen {
        width: 100%;
        height: 100%;
      }
      #container:-moz-full-screen {
        width: 100%;
        height: 100%;
        background-color: white;
      }
      #mainText {
        color: black;
      }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>テキストをフルスクリーン表示</h3>
    <div id="container">
    <div id="mainText">時は22世紀。人々はネコ型のロボットと暮らしていた。</div>
    </div>
    <form>
      <input type="button" value="全画面にする" id="setFS">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

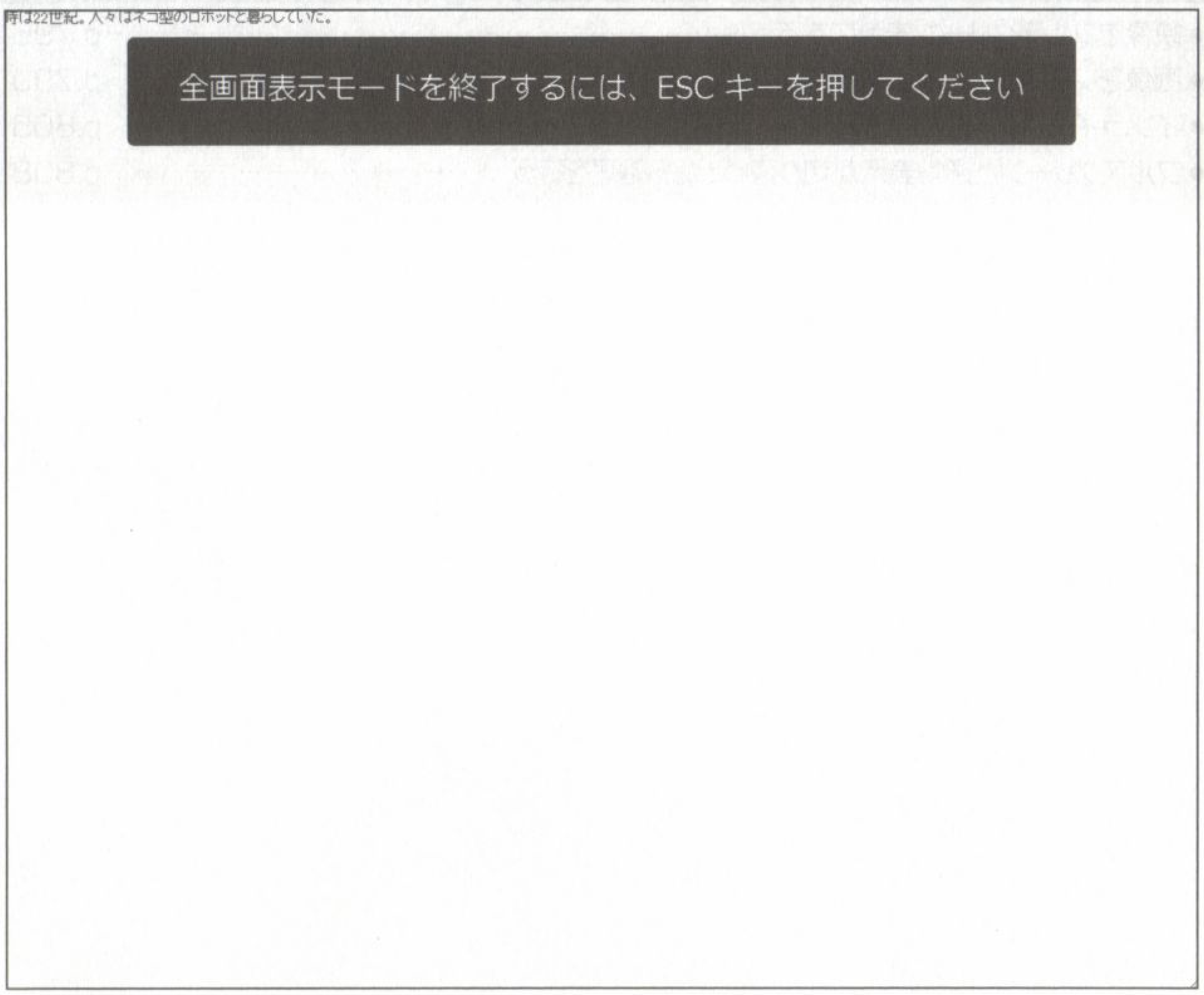
```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setFS").addEventListener("click", function(){
    var myText = document.getElementById("mainText");
```

```
    // Safari/Google Chrome用
    if(myText.webkitRequestFullScreen){
      myText.webkitRequestFullScreen();
    }
    // Firefox用
    if (myText.parentNode.mozRequestFullScreen){
      myText.parentNode.mozRequestFullScreen();
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「全画面にする」ボタンをクリックすると、テキストがフルスクリーン（全画面）で表示されます。



フルスクリーン表示の場合、最初に画面上に元に戻す方法が表示されます。escキーを押すと元の状態に戻ります。



ONEPOINT

テキストをフルスクリーンにするには テキストを囲んでいる要素の「requestFullscreen()」メソッドを使う

テキストをフルスクリーンにするには、テキストを囲んでいる要素の「requestFullscreen()」メソッドを使います。ただし、SafariやGoogle ChromeとFirefoxの両方で同じように動作させるには、スタイルシート、および、HTML要素に工夫が必要です。まず、Firefoxに対応させる場合には、テキストを内包している要素をさらに囲む親要素(「div」要素など)が必要になります。SafariやGoogle Chromeの場合は直接、テキストを内包している要素の「requestFullscreen()」メソッドを呼び出せばフルスクリーンになりますが、Firefoxではdiv要素などで囲み、さらにスタイルシートを設定する必要があります。Firefoxの場合は、テキストを内包している要素と、内包している親要素の両方にフルスクリーンモード時の表示位置やサイズなどを設定します。

COLUMN

Fullscreen APIのベンダープレフィックス

2012年現在、Fullscreen APIはベンダープレフィックス付きで使用することができます。また、仕様ではFullScreenのSが小文字になっていますが、ベンダープレフィックス付きのものは大文字になっています。SafariやGoogle ChromeなどのWebKit系では「webkitRequestFullScreen()」メソッド、Firefoxでは「mozRequestFullScreen()」メソッドを使ってフルスクリーンモードにすることができます。

関連項目▶▶▶

- 映像をフルスクリーン表示にする …… p.796
- 画像をフルスクリーン表示にする …… p.799
- インラインフレームをフルスクリーン表示にする …… p.805
- フルスクリーンと通常表示が切り替わったら処理を行う …… p.808

SECTION-341

インラインフレームをフルスクリーン表示にする

ここでは、ページ内に表示されているインラインフレームをフルスクリーン表示にする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

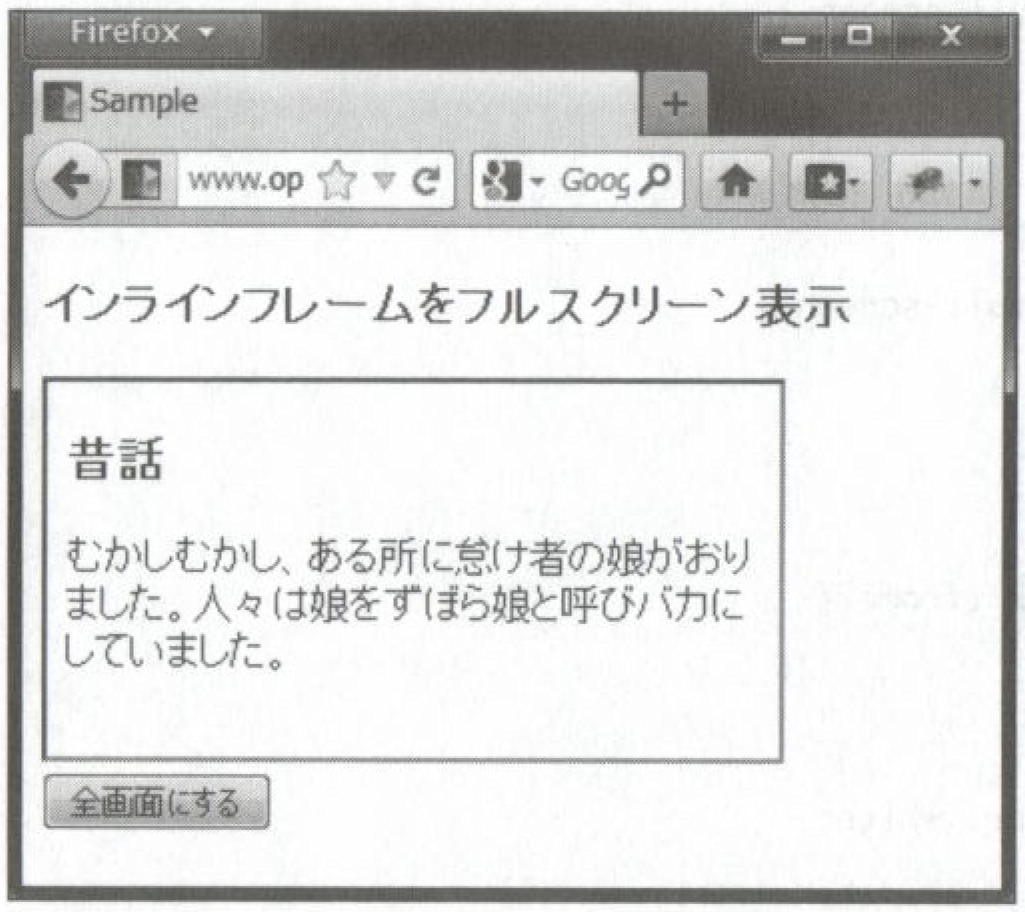
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      #story:-webkit-full-screen {
        width: 100%;
        height: 100%;
      }
      #container:-moz-full-screen {
        width: 100%;
        height: 100%;
      }
      :-moz-full-screen iframe {
        width: 100%;
        height: 100%;
        background-color: white;
        color: black;
      }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>インラインフレームをフルスクリーン表示</h3>
    <div id="container">
    <iframe src="text.html" id="story" allowfullscreen webkitallowfullscreen></iframe>
    </div>
    <form>
      <input type="button" value="全画面にする" id="setFS">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

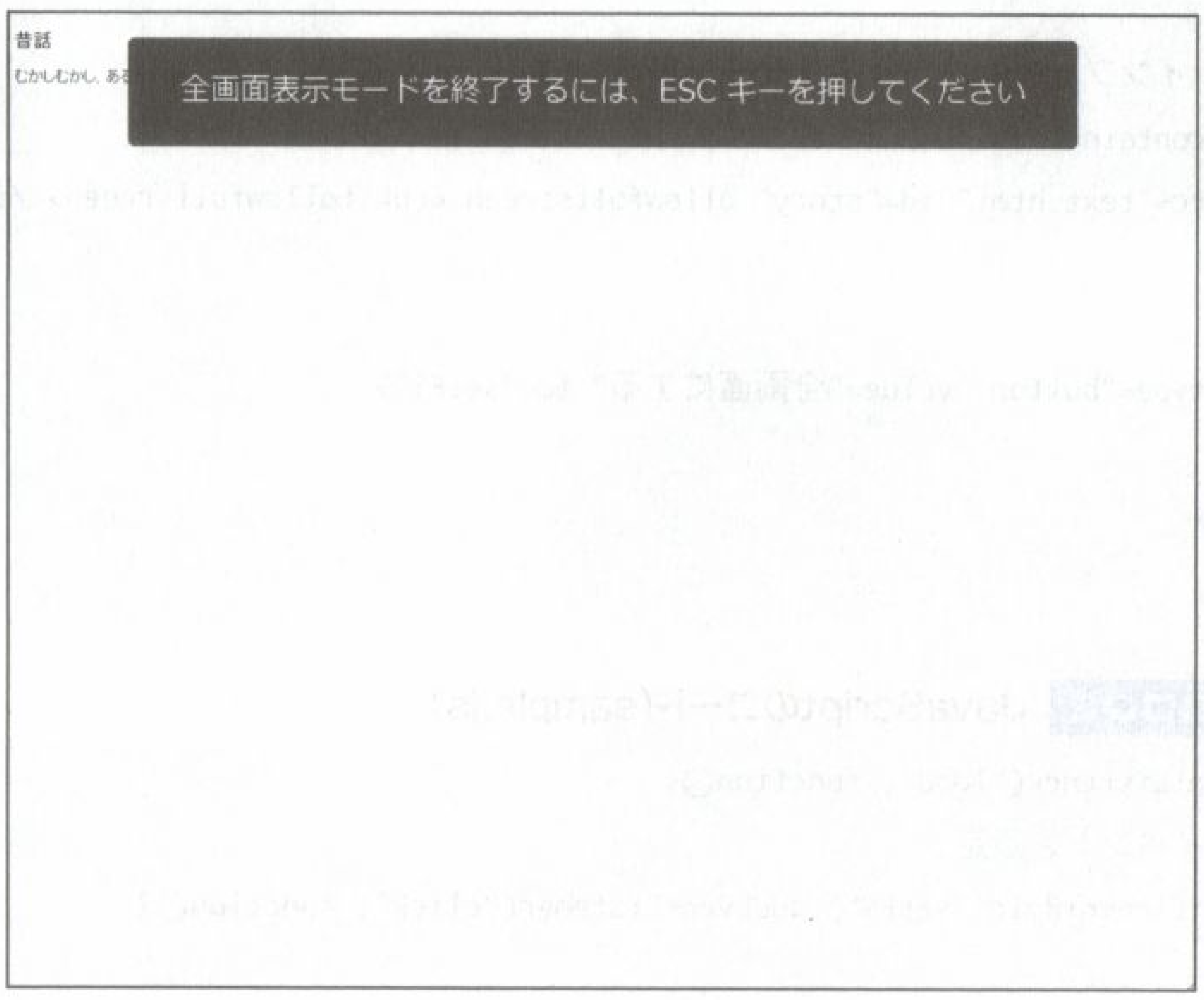
```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setFS").addEventListener("click", function(){
```

```
      var myIframe = document.getElementById("story");
      // Safari/Google Chrome用
      if(myIframe.webkitRequestFullScreen){
        myIframe.webkitRequestFullScreen();
      }
      // Firefox用
      if (myIframe.parentNode.mozRequestFullScreen){
        myIframe.parentNode.mozRequestFullScreen();
      }
    }, false);
  }, false);
```

ブラウザでindex.htmlを表示し、「全画面にする」ボタンをクリックすると、インラインフレームの内容がフルスクリーン（全画面）で表示されます。



フルスクリーン表示の場合、最初に画面上に元に戻す方法が表示されます。escキーを押すと元の状態に戻ります。なお、Google Chromeでは「webkitallowfullscreen」属性が設定されていないと、全画面表示するかどうかの許可ボタンが表示されます。



ONEPOINT

インラインフレームをフルスクリーンにするには「iframe」要素の「requestFullscreen()」メソッドを使う

インラインフレームをフルスクリーンにするには、「iframe」要素の「requestFullscreen()」メソッドを使います。ただし、SafariやGoogle ChromeとFirefoxの両方で同じように動作させるにはスタイルシート、および、HTML要素に工夫が必要です。まず、Firefoxに対応させる場合には、インラインフレームを囲む親要素（「div」要素など）が必要になります。SafariやGoogle Chromeの場合は直接、「iframe」要素の「requestFullscreen()」メソッドを呼び出せばフルスクリーンになりますが、Firefoxでは「div」要素などで囲み、さらにスタイルシートを設定する必要があります。Firefoxの場合は、「iframe」要素と、内包している親要素の両方にフルスクリーンモード時の表示位置やサイズなどを設定します。

また、Google Chromeでは「iframe」要素に「webkitallowfullscreen」属性を指定していない場合、フルスクリーン表示を許可するかどうかのボタンが表示されます。

COLUMN

Fullscreen APIのベンダープレフィックス

2012年現在、Fullscreen APIはベンダープレフィックス付きで使用することができます。また、仕様ではFullScreenのSが小文字になっていますが、ベンダープレフィックス付きのものは大文字になっています。SafariやGoogle ChromeなどのWebKit系では「webkitRequestFullScreen()」メソッド、Firefoxでは「mozRequestFullScreen()」メソッドを使ってフルスクリーンモードにすることができます。

関連項目 ▶▶▶

- 映像をフルスクリーン表示にする …… p.796
- 画像をフルスクリーン表示にする …… p.799
- テキストをフルスクリーン表示にする …… p.802
- フルスクリーンと通常表示が切り替わったら処理を行う …… p.808

SECTION-342

フルスクリーンと通常表示が切り替わったら処理を行う

ここでは、フルスクリーン表示と通常表示、通常表とフルスクリーン表示が切り替わったら処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Sample</title>
  <style>
   #sunflower:-webkit-full-screen {
     width: 720pt;
     height: 1080pt;
   }
   :-moz-full-screen #sunflower{
     width: 720pt;
     height: 1080pt;
   }
   #container:-moz-full-screen {
     display: table-cell;
     text-align: center;
     vertical-align: middle;
   }
  </style>
  <script src="js/sample.js"></script>
 </head>
 <body>
  <h3>フルスクリーン表示</h3>
  <div id="container">
  <img src="images/photo.jpg" id="sunflower" alt="ひまわり" width="72" height="108">
  </div>
  <form>
   <input type="button" value="全画面にする" id="setFS">
  </form>
 </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

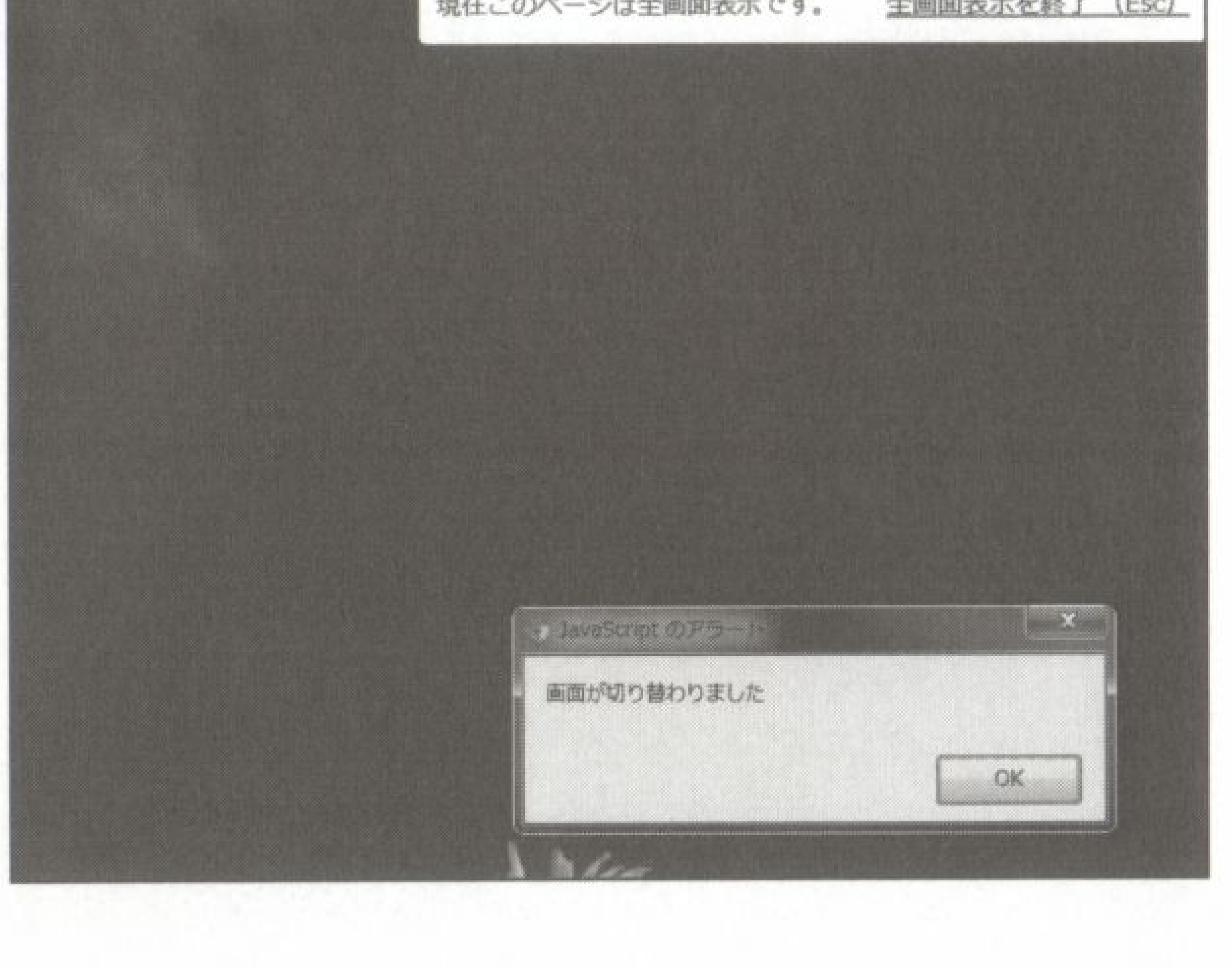
```
window.addEventListener("load", function(){
 // ボタンにイベントを設定
 document.getElementById("setFS").addEventListener("click", function(){
  var myImage = document.getElementById("sunflower");
```

```
    // Safari/Google Chrome用
    if(myImage.webkitRequestFullScreen){
      myImage.webkitRequestFullScreen();
    }
    // Firefox用
    if (myImage.mozRequestFullScreen){
      myImage.parentNode.mozRequestFullScreen();
    }
  }, false);
  // イベントハンドラを設定(Firefox用)
  document.addEventListener("mozfullscreenchange", function(){
    alert("画面が切り替わりました");
      }, false);
  // イベントハンドラを設定(Safari/Google Chrome用)
  document.addEventListener("webkitfullscreenchange", function(){
    alert("画面が切り替わりました");
      }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「全画面にする」ボタンをクリックすると、フルスクリーン（全画面）になります。



フルスクリーン表示に切り替わると、切り替わったことを示すアラートダイアログが表示されます。また、通常の画面に戻ったときにも、切り替わったことを示すアラートダイアログが表示されます。



ONEPOINT 画面切り替え時に処理を行うには「fullscreenchange」イベントを捕捉する

フルスクリーン表示に切り替わった場合、もしくは、フルスクリーン表示から通常表示に切り替わった場合に処理を行うには、「document」オブジェクトの「fullscreenchange」イベントを捕捉します。2012年7月時点では、Firefoxは「mozfullscreenchange」イベント、Google Chrome/Safariは「webkitfullscreenchange」イベントとしてイベント名が定義されています。

また、Firefoxでは、フルスクリーン表示になっているかどうかは、「document.mozFullScreen」プロパティで確認することができます。「true」であればフルスクリーン表示、「false」なら通常表示になります。

関連項目 ▶▶▶

- イベントハンドラを設定する …… p.396
- イベントリスナーを設定する …… p.397
- 映像をフルスクリーン表示にする …… p.796
- 画像をフルスクリーン表示にする …… p.799
- テキストをフルスクリーン表示にする …… p.802
- インラインフレームをフルスクリーン表示にする …… p.805

SECTION-343

バイブレーション機能を使う

ここでは、バイブレーション機能を使う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>バイブレーション／振動</h3>
    <form>
      <input type="button" value="3秒振動" id="vib1">
      <input type="button" value="3回振動(2秒振動、1秒停止の繰り返し)" id="vib2">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタン1にイベントを設定
  document.getElementById("vib1").addEventListener("click", function(){
    navigator.mozVibrate(3000);
  }, false);
  // ボタン2にイベントを設定
  document.getElementById("vib2").addEventListener("click", function(){
    navigator.mozVibrate([1000, 1000, 1000, 1000, 1000]);
  }, false);
}, false);
```

HINT

バイブレーション機能は、ノート型やデスクトップ型のパソコンではなく、スマートフォンが対象です。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、ボタンに応じてデバイスが振動します。



ONEPOINT　バイブレーション機能を使うにはVibration APIを使う

バイブレーション機能を使うには、Vibration APIを使います。このAPIは、「navigator.vibrate()」メソッドを呼び出すことで利用できます。ただし、Firefoxでは「navigator.vibrate()」とすると、1回、振動するだけです。仕様に沿った動作をするのは「navigator.mozVibrate()」メソッドで、パラメータが数値の場合、指定したミリ秒だけ振動させます。「2000」を指定すれば、2秒間、振動することになります。

パラメータに配列を指定した場合、配列内の数値に応じて振動させます。最初の要素が振動するミリ秒で、次の要素が振動を止めるミリ秒になります。配列要素の数だけ、振動と停止を繰り返すことになります。

SECTION-344

バッテリーの充電レベルを取得する

ここでは、スマートフォンやノート型パソコンなどのバッテリーの充電レベルを取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>バッテリ状態を表示</h3>
    <form>
      <input type="button" value="バッテリーチェック" id="check">
    </form>
    <div>充電レベル：<output></output></div>
  </body>
</html>
```

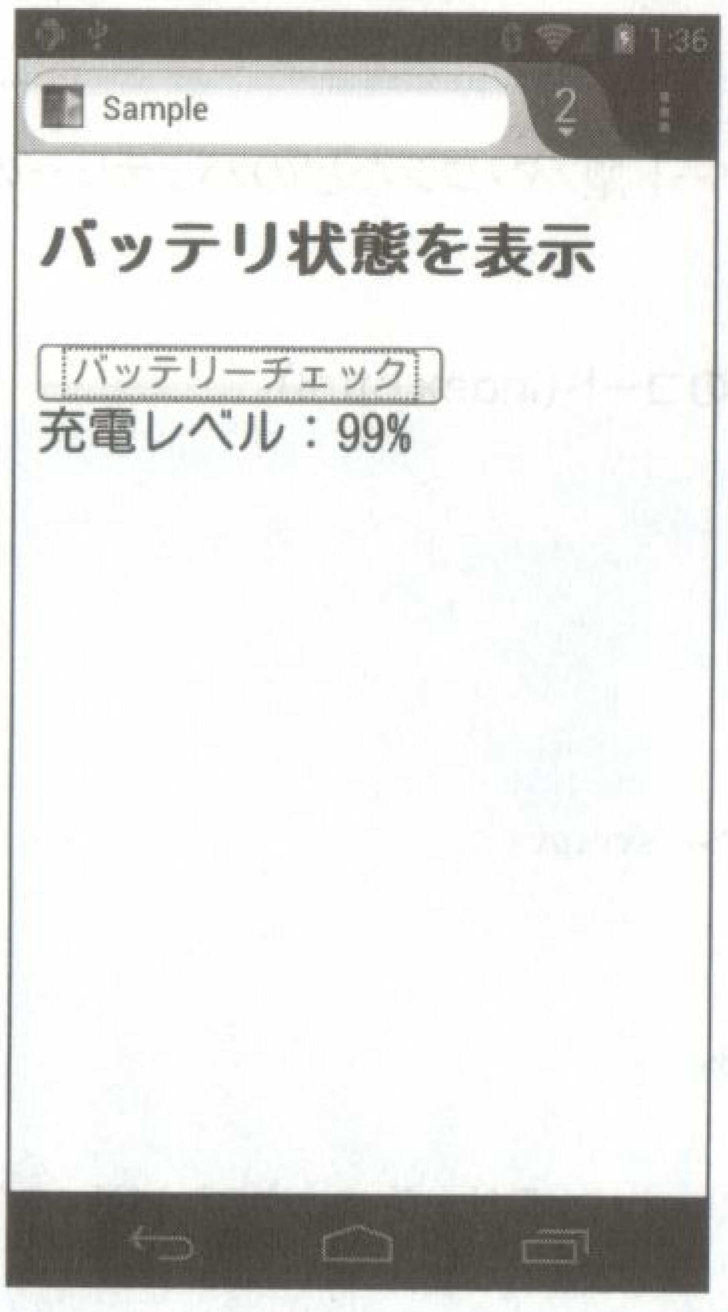
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // バッテリーの状態を入れたオブジェクトを読み出し
  var battery = navigator.battery || navigator.mozBattery;
  if (!battery){
    ele.innerHTML = "Battery Status APIに対応していません";
  }
  // ボタンにイベントを設定
  document.getElementById("check").addEventListener("click", function(){
    ele.innerHTML = (battery.level * 100 ) +"%";
  }, false);
}, false);
```

HINT

この機能は、スマートフォンやタブレット、ノートパソコンが対象です(デスクトップパソコンでも情報を取得することはできる)。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、現在のバッテリーの充電レベルが表示されます。



ONEPOINT

バッテリーの充電レベルを取得するには「battery」オブジェクトの「level」プロパティを使う

スマートフォンやノート型パソコンのバッテリー状態に関する情報は、「navigator.battery」オブジェクトに格納されています。ベンダープレフィックス付きでないと動作しない場合もあります。

充電レベルは、「battery」オブジェクトの「level」プロパティを参照することで取得することができます。「1.0」であれば100%充電、「0.25」なら25%となります。

関連項目 ▶▶▶

- バッテリーの充電状態を取得する ……… p.815
- バッテリーの充電状態が変化したら処理を行う ……… p.817
- バッテリーレベルが変化した場合に処理する ……… p.819

SECTION-345

バッテリーの充電状態を取得する

ここでは、スマートフォンやノート型パソコンなどのバッテリーの充電状態を取得する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>充電状態を表示</h3>
    <form>
      <input type="button" value="充電状態をチェック" id="check">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // バッテリーの状態を入れたオブジェクトを読み出し
  var battery = navigator.battery || navigator.mozBattery;
  if (!battery){
    ele.innerHTML = "Battery Status APIに対応していません";
  }
  // ボタンにイベントを設定
  document.getElementById("check").addEventListener("click", function(){
    // 充電中か調べる
    if (battery.charging){
      ele.innerHTML = "充電中です";
    }else{
      ele.innerHTML = "充電中していません";
    }
  }, false);
}, false);
```

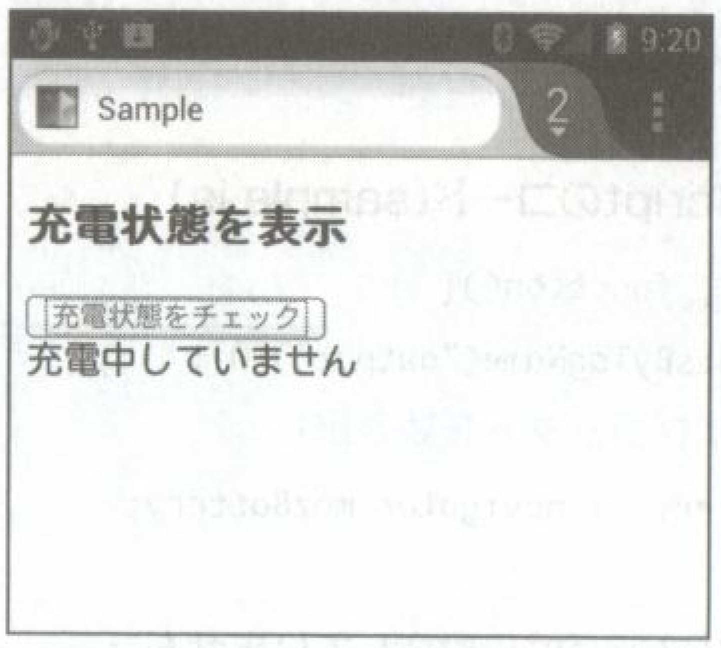
HINT

この機能は、スマートフォンやタブレット、ノートパソコンが対象です(デスクトップパソコンでも情報を取得することはできる)。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、現在のバッテリーの充電レベルが表示されます。充電中の場合は、次のように表示されます。



充電中ではない場合は、次のように表示されます。



ONEPOINT

バッテリーの充電状態を取得するには「battery」オブジェクトの「charging」プロパティを使う

スマートフォンやノート型パソコンのバッテリー状態に関する情報は、「navigator.battery」オブジェクトに格納されています。ベンダープレフィックス付きでないと動作しない場合もあります。

充電状態は、「battery」オブジェクトの「charging」プロパティを参照することで取得することができます。「true」であれば充電中、「false」であれば充電中ではないことを示します。

関連項目 ▶▶▶

● バッテリーの充電レベルを取得する ……… p.813
● バッテリーの充電状態が変化したら処理を行う ……… p.817
● バッテリーレベルが変化した場合に処理する ……… p.819

SECTION-346

バッテリーの充電状態が変化したら処理を行う

ここでは、スマートフォンやノート型パソコンなどのバッテリーの充電状態が変化した場合の方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>充電状況を監視</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // バッテリーの状態を入れたオブジェクトを読み出し
  var battery = navigator.battery || navigator.mozBattery;
  if (!battery){
    ele.innerHTML = "Battery Status APIに対応していません";
  }
  // イベントを設定
  battery.onchargingchange = function(){
    ele.innerHTML = "充電状態が変化しました<br>";
    if (battery.charging){
      ele.innerHTML += "充電を開始しました";
    }else{
      ele.innerHTML += "充電を終了しました";
    }
  }
}, false);
```

HINT

この機能は、スマートフォンやタブレット、ノートパソコンが対象です。

ブラウザでindex.htmlを表示し、充電を開始したり、充電を停止したりすると、メッセージが表示されます。



ONEPOINT 充電状態が変化した場合に処理を行うには「chargingchange」イベントを使う

バッテリーが充電を開始した場合や、充電を中止した場合に処理を行うには、「chargingchange」イベントを検知します。「onchargingchange」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントリスナーとして設定します。

充電が行われたかどうかは、「charging」プロパティで調べることができます。「charging」プロパティが「true」なら充電中、「false」なら充電が中止された(充電のためのケーブルが外されたなど)ことになります。

関連項目 ▶▶▶

- イベントハンドラを設定する ………… p.396
- イベントリスナーを設定する ………… p.398
- バッテリーの充電レベルを取得する ………… p.813
- バッテリーの充電状態を取得する ………… p.815
- バッテリーレベルが変化した場合に処理する ………… p.819

SECTION-347

バッテリーレベルが変化した場合に処理する

ここでは、スマートフォンやノート型パソコンなどのバッテリーレベルが変化した場合に処理する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>充電レベルが変化した場合に処理</h3>
    <output></output>
  </body>
</html>
```

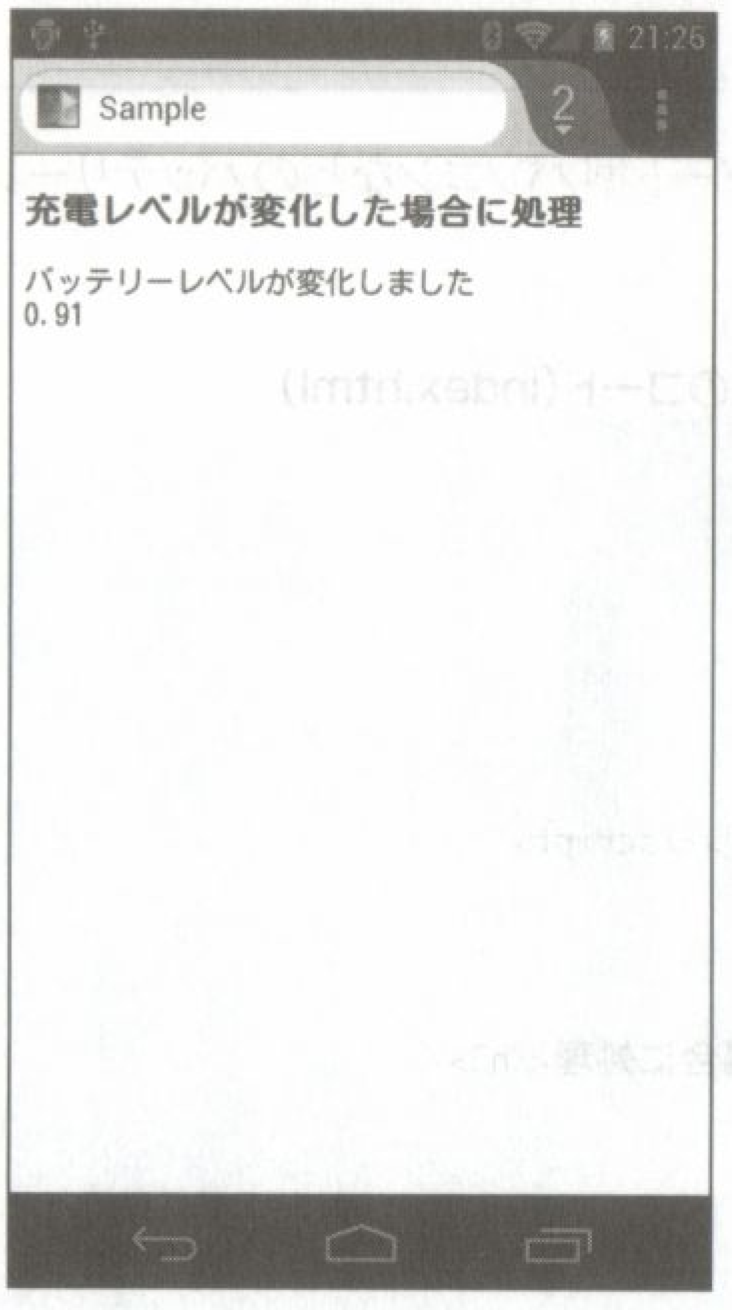
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // バッテリーの状態を入れたオブジェクトを読み出し
  var battery = navigator.battery || navigator.mozBattery;
  if (!battery){
    ele.innerHTML = "Battery Status APIに対応していません";
  }
  // イベントを設定
  battery.addEventListener("levelchange", function(){
    // バッテリーレベルを表示
    var n = battery.level;
    ele.innerHTML += "バッテリーレベルが変化しました<br>"+n;
  }, false);
}, false);
```

HINT

この機能は、スマートフォンやタブレット、ノートパソコンが対象です。

ブラウザでindex.htmlを表示し、バッテリーレベルが変化すると、メッセージと充電レベルが表示されます。



ONEPOINT **バッテリーレベルが変化した場合に処理するには「levelchange」イベントを使う**

バッテリーレベル（充電レベル）が変化すると、「levelchange」イベントが発生します。「battery」オブジェクトの「onlevelchange」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントリスナーとして設定します。

また、バッテリーレベルは、「battery」オブジェクトの「level」プロパティを参照することで取得することができます。

関連項目 ▶▶▶

- イベントハンドラを設定する …… p.396
- イベントリスナーを設定する …… p.398
- バッテリーの充電レベルを取得する …… p.813
- バッテリーの充電状態を取得する …… p.815
- バッテリーの充電状態が変化したら処理を行う …… p.817

SECTION-348

ゲームパッドが使用可能か調べる

ここでは、ブラウザで利用できるゲームパッドが使用可能か調べる方法について解説します。

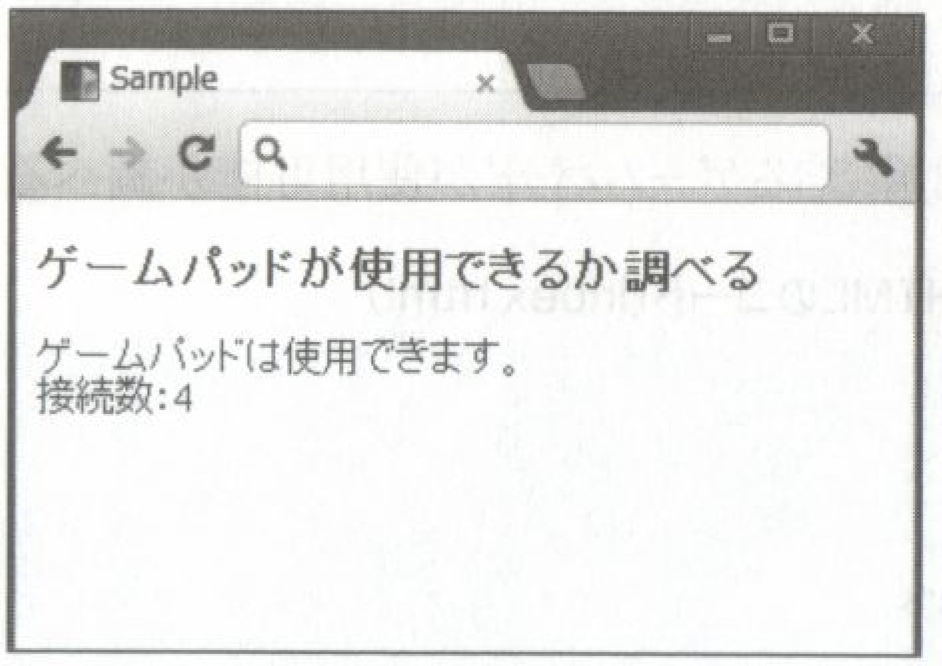
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>ゲームパッドが使用できるか調べる</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // GamePadオブジェクト (Google Chromeの場合)
  navigator.gamepads = navigator.gamepads || navigator.webkitGamepads;
  // ゲームパッドが使用できるか調べる
  if (navigator.gamepads){
    // 使用可能な場合の処理
    ele.innerHTML = "ゲームパッドは使用できます。<br>";
    // 接続されているパッドの数を取得
    var n = navigator.gamepads.length;
    ele.innerHTML += "接続数 :"+n;
  }else{
    ele.innerHTML = "ゲームパッドは接続されていません";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、ゲームパッドが利用可能な場合は、その旨メッセージとゲームパッドの接続数が表示されます。



ONEPOINT ゲームパッドを使用可能か調べるには「gamepads」オブジェクトを使う

ゲームパッドを利用するには、「navigator」オブジェクトの「gamepads」オブジェクトの有無を調べます。「gamepads」オブジェクトがあれば利用可能、そうでなければ利用できないことになります。「gamepads」オブジェクトには、それぞれ接続されたゲームパッドのオブジェクトが配列で格納されています。このため、「gamepads[0]」とすると、接続されているゲームパッドその1の情報にアクセスすることができます。

プロパティ	説明
id	ゲームパッド名/ベンダー名
index	インデックス番号
timestamp	入力時の時間(ミリ秒)
buttons	ボタン(配列の数=入力可能なボタン数)
axes	スティックの入力方向を示す配列

COLUMN ゲームパッドを有効にするには

Google Chrome 21では、「gamepads」オブジェクトは「webkitGamepads」オブジェクトになっています。また、デフォルトではゲームパッドは使用できないようになっているため、アドレスバーに「about:flags」と入力し、表示される設定画面でゲームパッドを有効にする必要があります。

関連項目 ▶▶▶

● バイブレーション機能を使う ……………………………… p.811

SECTION-349

要素内の文字を編集可能にする

ここでは、ページ内に表示されている要素内の文字を編集可能にする方法について解説します。

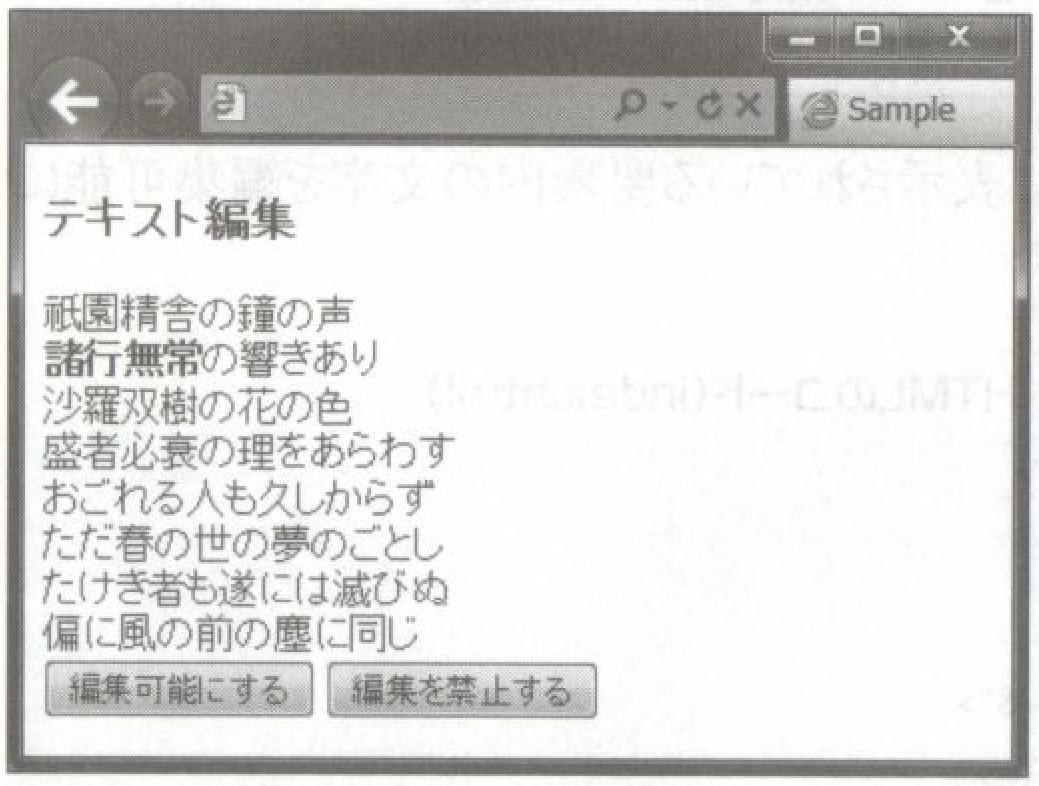
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>テキスト編集</h3>
    <div id="editArea">祇園精舎の鐘の声<br><b>諸行無常</b>の響きあり<br>
    沙羅双樹の花の色<br>盛者必衰の理をあらわす<br>おごれる人も久しからず<br>
    ただ春の世の夢のごとし<br>たけき者も遂には滅びぬ<br>偏に風の前の塵に同じ
    </div>
    <form>
      <input type="button" value="編集可能にする" id="startEdit">
      <input type="button" value="編集を禁止する" id="endEdit">
    </form>
  </body>
</html>
```

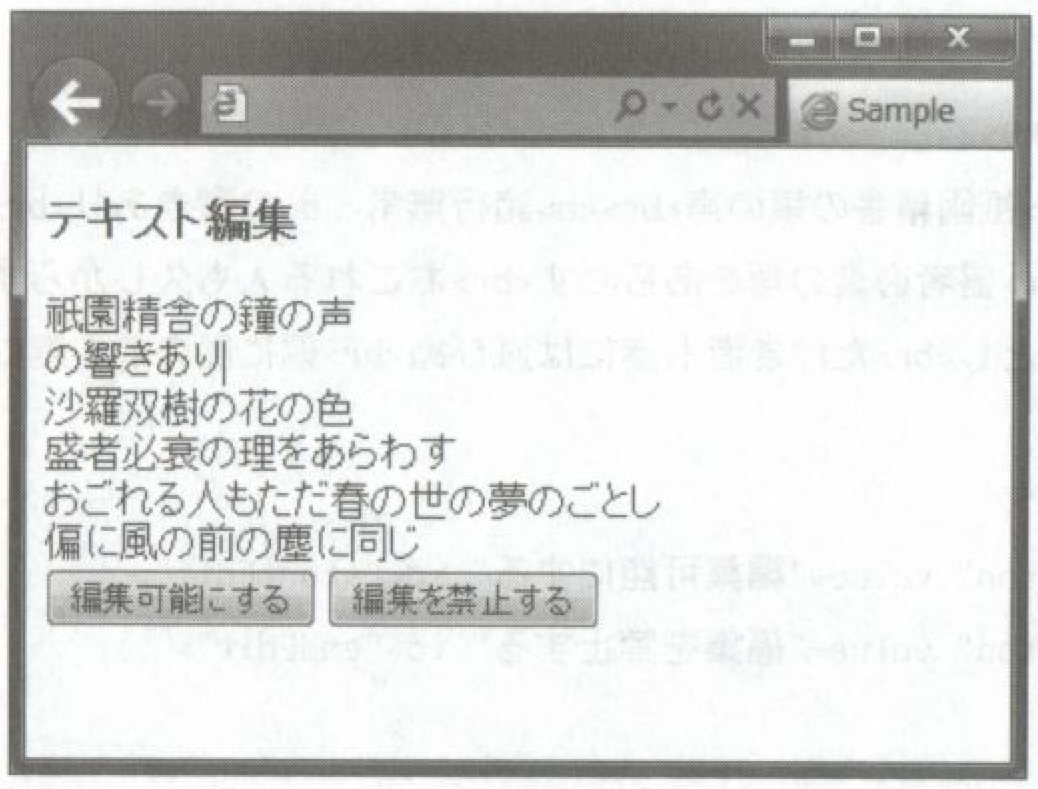
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // 編集可能領域
  var edit = document.getElementById("editArea");
  // 編集可能ボタンにイベントを設定
  document.getElementById("startEdit").addEventListener("click", function(){
    edit.contentEditable = true;
  }, false);
  // 編集禁止ボタンにイベントを設定
  document.getElementById("endEdit").addEventListener("click", function(){
    edit.contentEditable = false;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「編集可能にする」ボタンをクリックすると、指定した要素の文字が編集可能になります。



文字を追加・削除した後に「編集を禁止する」ボタンをクリックすると、編集ができなくなります。



ONEPOINT 要素内の文字や画像を編集するには「contentEditable」プロパティを使う

要素内の文字や画像を編集するには、要素の「contentEditable」プロパティに「true」を指定します。「true」を指定すると、その要素内にある文字や画像は編集可能になり、文字の追加や削除が可能になります。また、子要素や子孫要素も、親要素と同様に編集可能になります。

また、スクリプトで「contentEditable」プロパティを操作する方法以外に、要素に「contenteditable」属性を指定しても、編集が可能になります。

WebSocketが使用可能か調べる

ここでは、サーバーとソケット通信ができるWebSocketが使用可能か調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>WebSocketが使用できるか調べる</h3>
    <output></output>
  </body>
</html>
```

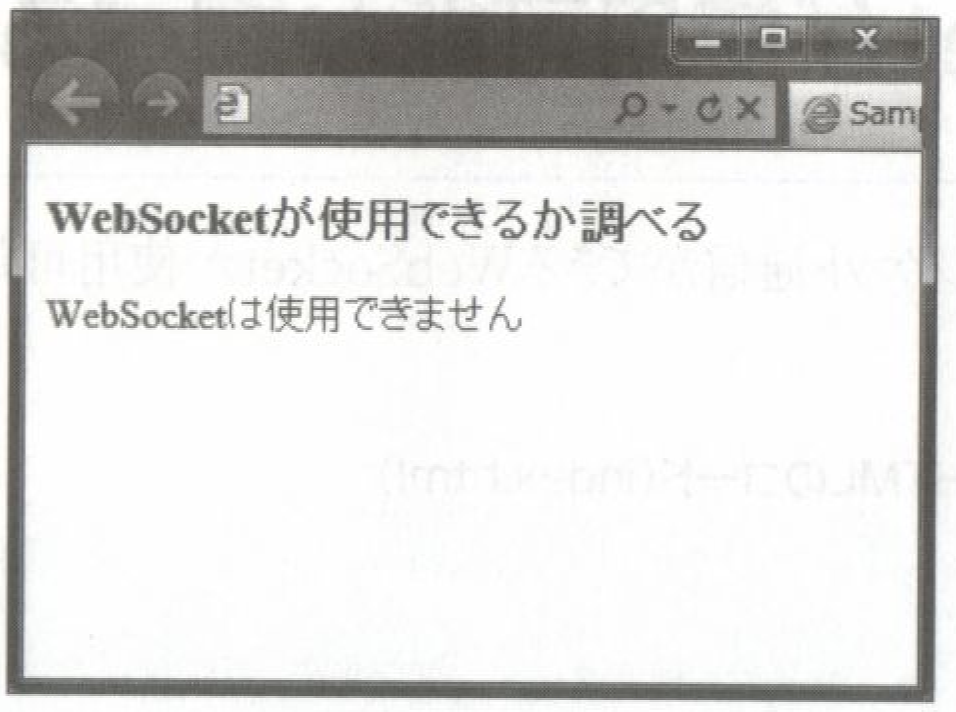
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WebSocketオブジェクトが使用できるか調べる
  if (window.WebSocket){
    // 使用可能な場合の処理
    ele.innerHTML = "WebSocketは使用できます。";
  }else{
    ele.innerHTML = "WebSocketは使用できません";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、WebSocketが使用できる場合は、次のように表示されます。



WebSocketが使用できない場合は、次のように表示されます。



ONEPOINT **WebSocketが利用できるかどうかは「WebSocket」オブジェクトの有無を調べる**

JavaScriptでは、サーバーと通信する手段がいくつか用意されています。多く利用されているのが、非同期通信を使ったものです。これは、「XMLHttpRequest」オブジェクトを利用して通信を行います。ただし、チャットやゲームなどでリアルタイムにデータをやり取りする用途には向いていません。

リアルタイムにサーバーとやり取りする方法として、WebSocketがあります。WebSocketは、その名の通り、ソケット通信を行います。これはサーバーとクライアント側双方向で通信ができるため、チャットやゲームに利用するには便利です。

WebSocketが使えるかどうかは、「window」オブジェクト内に「WebSocket」オブジェクトがあるかどうかを調べます。なお、「WebSocket」オブジェクトがない場合は、「undefined」になります。

関連項目 ▶▶▶

● WebSocketでサーバーに接続する……p.827
● WebSocketを使ってサーバーとデータをやり取りする……p.829

SECTION-351

WebSocketでサーバーに接続する

ここでは、WebSocketでサーバーに接続する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>WebSocketで接続する</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WebSocketオブジェクトを生成する
  var ws = new WebSocket("ws://192.168.11.1:9001");
  // 接続状態を確認
  var msg = ["接続中", "接続済み", "切断中", "切断された"];
  // 接続状態を表示する
  ele.innerHTML = "ステータス:"+msg[ws.readyState];
}, false);
```

ブラウザでindex.htmlを表示すると、WebSocketでサーバーに接続中の場合は、次のように表示されます。



ONEPOINT サーバーに接続するには「new WebSocket()」とする

WebSocketでサーバーに接続するには、「new WebSocket()」として、パラメータに接続先のサーバーを指定します。このときには、「http://」ではなく、「ws://」を使います。データを暗号してセキュアな通信を行う場合には、「wss://」を指定します。また、「192.168.11.1:4649」のようにポート番号を指定することもできます。他にも、「WebSocket()」の2番目のパラメータにサブプロトコルを指定することもできます。

サーバーとの接続状況は、「WebSocket」オブジェクトの「readState」プロパティで確認することができます。「readState」プロパティと状態の関係は、次の表のようになります。

値	キーワード値	説明
0	CONNECTING	接続中
1	OPEN	接続済み
2	CLOSING	切断中
3	CLOSED	切断された

また、サーバーとの接続が確立されると、「open」イベントが発生します。イベント処理に関しては、396ページや398ページを参照してください。

関連項目 ▶▶▶

- WebSocketが使用可能か調べる……………………p.825
- WebSocketを使ってサーバーとデータをやり取りする……………………p.829

SECTION-352

WebSocketを使って サーバーとデータをやり取りする

ここでは、WebSocketでサーバーとデータをやり取りする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>データを送受信する</h3>
    <form>
      文字：<input type="text" value="Furuhata" id="uName"><br>
      <input type="button" value="データ送信" id="check">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // WebSocketオブジェクトを生成する
  var ws = new WebSocket("ws://192.168.11.1:9002");
  // 受信したデータを表示
  ws.onmessage = function(msg){
    ele.innerHTML += "サーバーから："+msg.data+"<br>";
  }
  // 接続時のサーバーからのメッセージを表示
  ws.onopen = function(data){
    ele.innerHTML = "接続完了イベント発生<br>";
  }
  document.getElementById("check").addEventListener("click", function(){
    // フォームに入力した文字を読み出す
    var text = document.getElementById("uName").value;
    // サーバーに入力した文字を送信
    ws.send(text);
  }, false);
}, false);
```

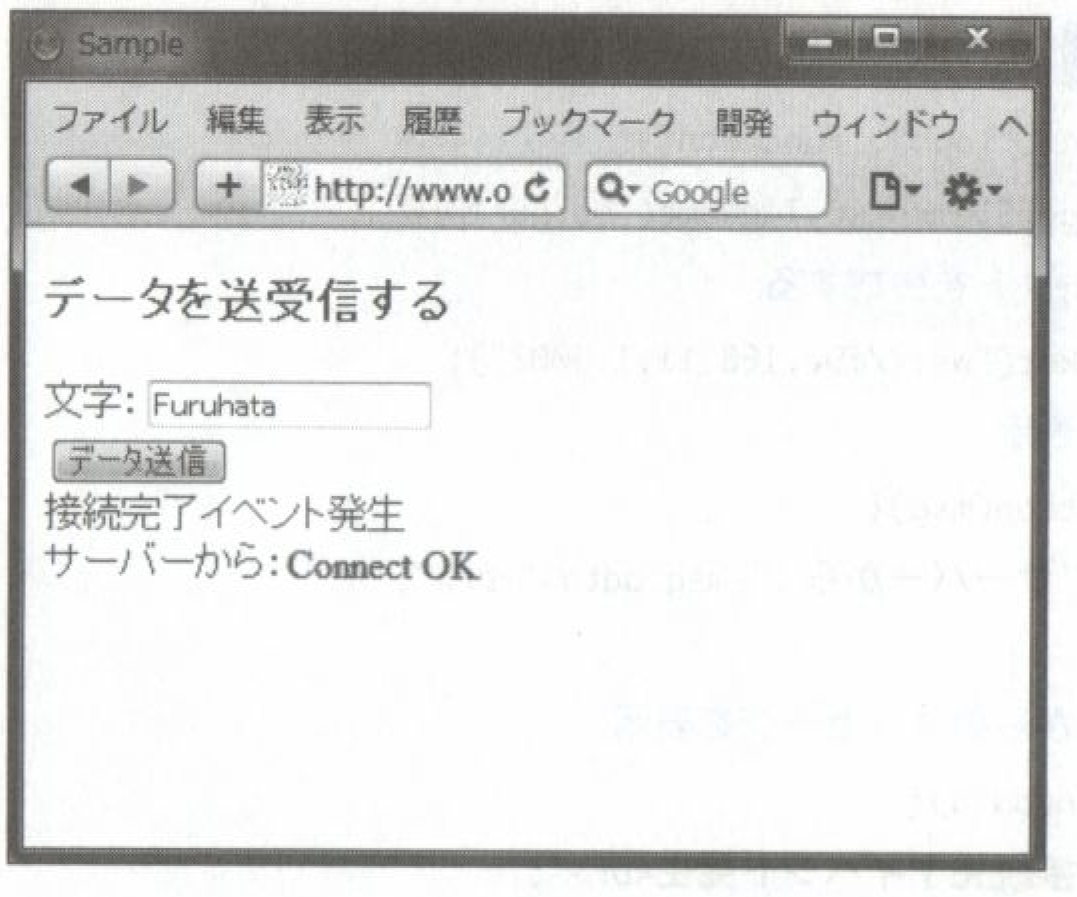
SAMPLE CODE サーバー側のコード(ws.js)

```
var ws = require( 'websocket-server');
// サーバー生成
var server = ws.createServer();
// 接続時のイベント
server.addListener('connection',function(conn){
  // 接続時にメッセージを送信
  conn.send("Connect OK");
  console.log("Connect.......");
  // メッセージ受信時のイベント
  conn.addListener('message', function(message){
    // メッセージを送信
    conn.send("Length = "+message.length);
    console.log("Send data");
  });
});
server.listen(9002);
```

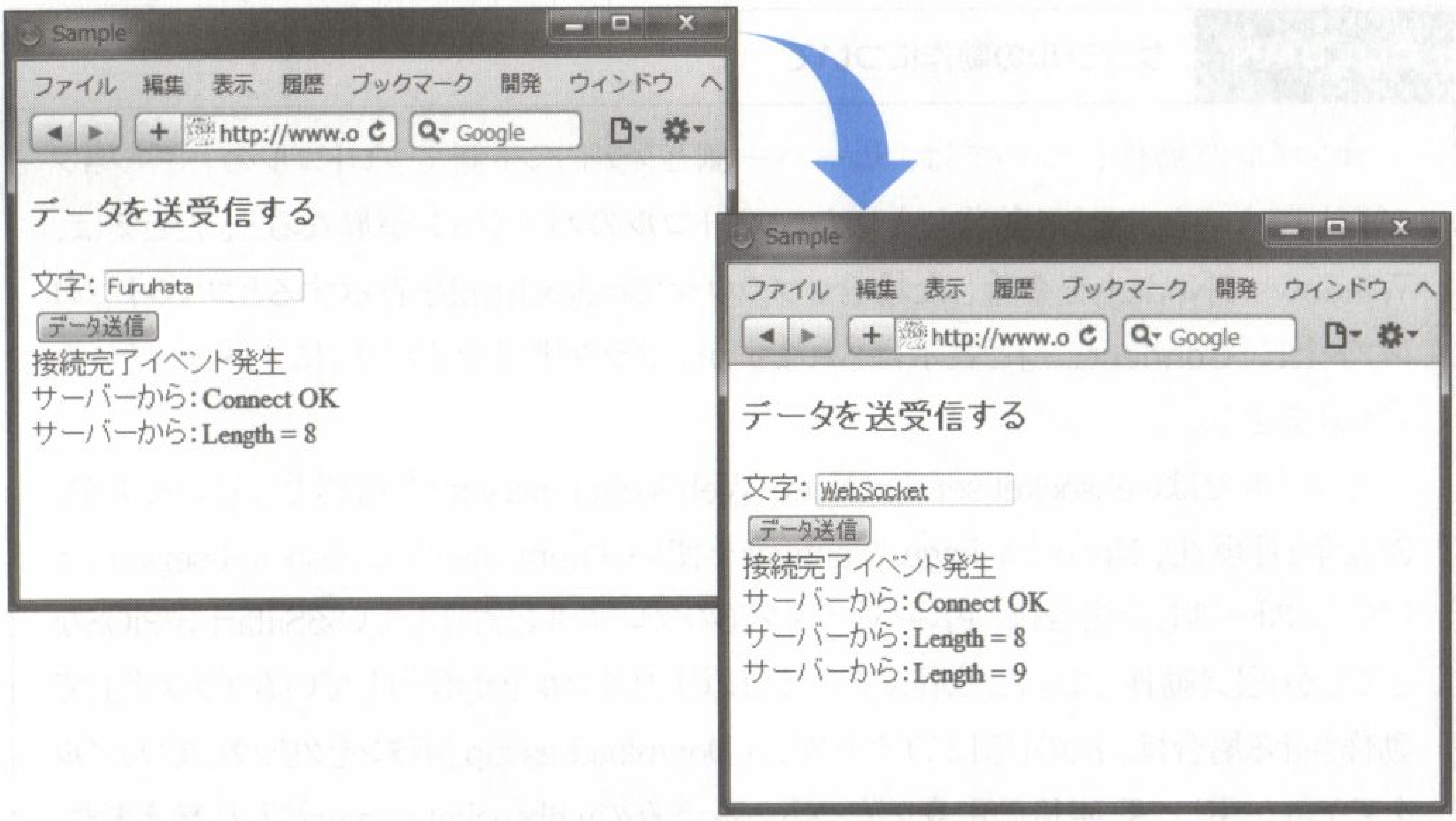
HINT

サーバー側には、node.jsを利用しています。

サーバー側でws.jsを実行した後に、ブラウザでindex.htmlを表示してサーバーに接続すると、サーバーからのメッセージが表示されます。



テキストフィールドに文字を入力し、「データ送信」ボタンをクリックすると、文字がサーバーに送信されます。その後、文字の長さを計算した結果がサーバーから返され、表示されます。



ONEPOINT サーバーとやり取りするにはメッセージ送信を使う

WebSocketを使ってサーバーとやり取りするには、メッセージ送信を使って行います。サーバーから接続された場合やデータが送信されると、クライアント側には「message」イベントが発生します。クライアント側では、「WebSocket」オブジェクトの「onmessage」プロパティにイベントハンドラを設定するか、「addEventListener()」メソッドを使ってイベントリスナーを使って設定します。

イベントハンドラには、サーバーから送信されたデータを格納したオブジェクトが渡されます。このオブジェクトの「data」プロパティにサーバーから送信されたメッセージが入ります。

クライアント側からサーバーにデータを送信する場合は、「WebSocket」オブジェクトの「send()」メソッドを使います。パラメータには、送信するデータ(テキストデータ)を指定します。

サーバー側で処理した結果が返された場合、「message」イベントが発生するので、状況に応じて処理を行います。

COLUMN Node.jsとは

Node.jsとは、PHPのようにサーバー側で動作するプログラムです。特長としてはサーバー側のプログラムをJavaScriptで記述できることです。Node.jsでWebSocketsを使用する場合は、exprerssやwebsocket-server(Node WebSocket Server)などを追加インストールする必要があります。Node.jsについては、次のページを参照してください。

URL http://nodejs.org/

COLUMN サンプルの動作について

サンプルの動作については、サーバー側とクライアント側でプロトコルのバージョンが同じでないと、正しく動作しません。プロトコルのバージョンが異なると、たとえば、WindowsでNode.jsを準備した場合、ブラウザでindex.htmlを表示するとコマンドプロンプト上に「Connect.......」と表示はされますが、ブラウザ（クライアント）にメッセージが表示されません。

サンプルではwebsocket-server（Node WebSocket Server）を前提としていますが、2012年7月現在、Node Package Managerを使い、「npm install websocket-server」としてインストールした場合は、古いバージョン76のプロトコルに対応しているSafari 5やiOSなどでしか正しく動作しません。新しいバージョンのプロトコルをサポートしているブラウザ上で動作させる場合は、次のURLにアクセスし、「Download as zip」ボタンをクリックしてファイルをダウンロードし、解凍後の中身をインストール済みのwebsocket-serverと入れ替えます。そうすることで、Google Chrome 21やSafari 6、IE10でも動作するようになります（ただし、Firefox 14では動作しない）。なお、websocket-serverは、Windows 7でNode.jsをインストールしている場合は「C:¥Users¥<ユーザー名>¥node_modules¥websocket-server」にインストールされているので、「websocket-server」の中身を入れ替えてください。

URL https://github.com/VanCoding/node-websocket-server/downloads

関連項目 ▶▶▶

● WebSocketが使用可能か調べる……p.825
● WebSocketでサーバーに接続する……p.827

SECTION-353

Indexed Databaseが使用可能か調べる

ここでは、クライアント側でデータベースが使用可能か調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Indexed DBが使用できるか調べる</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Indexed DBオブジェクトが使用できるか調べる
  var db = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB;
  if(db){
    ele.innerHTML = "Indexed Databaseは使用できます";
  }else{
    ele.innerHTML = "Indexed Databaseは使用できません";
  }
}, false);
```

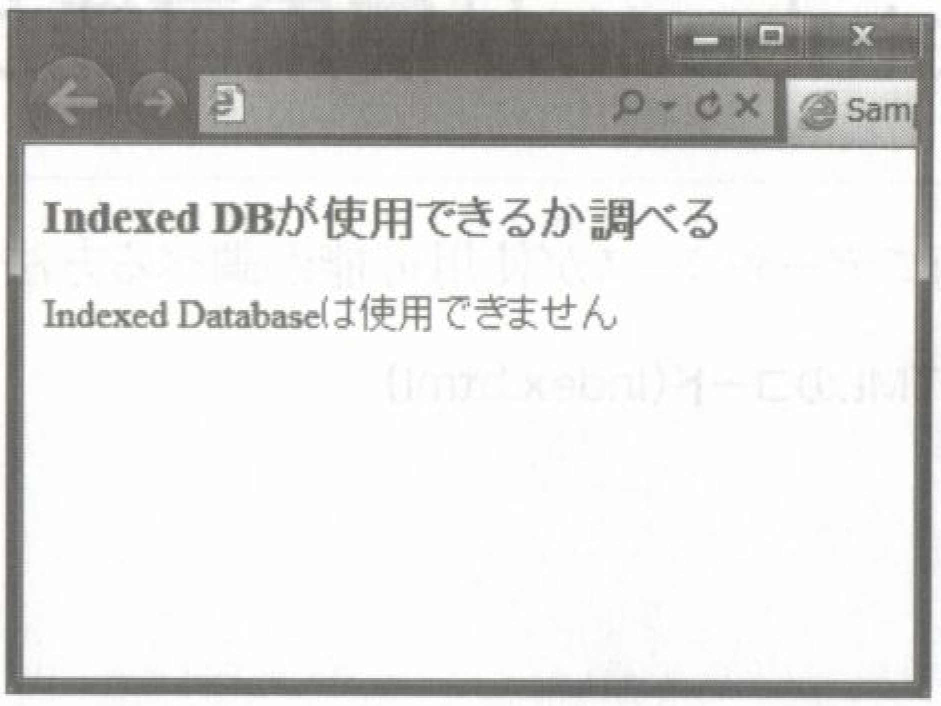
HINT

FirefoxでもIndexed Databaseをサポートしていますが、本書のサンプルは動作しません。

ブラウザでindex.htmlを表示すると、Indexed Databaseが使用できる場合は、次のように表示されます。



Indexed Databaseが使用できない場合は、次のように表示されます。



ONEPOINT

Indexed Databaseが使用できるかどうかは「IndexedDB」オブジェクトの有無を調べる

データベースはサーバー側で持つのが一般的です。データベースは、SQLを利用したものか、KVS(Key Value Store)を利用しています。HTML5では、クライアント側のデータベースとして、Indexed Databaseを定義しています。これはSQLではなく、KVS方式でデータを格納するタイプです。以前はSQLをベースにしたWeb SQLがありましたが、現在では仕様から外されています。ただし、iOS 5やAndroid 4までは利用することが可能です。

Indexed Databaseが使用できるかどうかは、「window」オブジェクト内に「IndexedDB」オブジェクトがあるかどうかを調べます。「IndexedDB」オブジェクトがない場合は、「undefined」が返されます。なお、2012年7月時点ではベンダープレフィックスを付けないと利用できません。このため、「IDBTransaction」オブジェクトや「IDBKeyRange」オブジェクトも、次のように記述する必要があります。

```
var IDBTransaction =
window.IDBTransaction || window.webkitIDBTransaction || window.mozIDBTransaction;
var IDBKeyRange =
window.IDBKeyRange || window.webkitIDBKeyRange || window.mozIDBKeyRange;
```

関連項目 ▶▶▶

SECTION-354

Indexed Databaseでデータベースを開く

ここでは、Indexed Databaseでデータベースを開く方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>データベースを開く</h3>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Indexed DBオブジェクトが使用できるか調べる
  var db = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB;
  if(!db){
    ele.innerHTML = "Indexed Databaseは使用できません";
    return;
  }
  // データベースを開く(もしくは新規作成)。データベースバージョンは1
  var req = db.open("testDB001", 1);
  // データベースのオープンに成功したときの処理
  req.onsuccess = function(evt){
    // 開いたデータベースを取得(IDBDatabaseオブジェクト)
    var myDB = this.result;
    // 結果を表示
    ele.innerHTML = "データベースをオープンしました<br>";
    ele.innerHTML += "データベース名:"+myDB.name+"<br>";
    ele.innerHTML += "バージョン:"+myDB.version;
  }
  // データベースのオープンに失敗したときの処理
  req.onerror = function(evt){
    ele.innerHTML = "データベースのオープンに失敗しました";
  }
}, false);
```

ブラウザでindex.htmlを表示し、データベースが正常に開くことができた場合は、データベース名とバージョン名が表示されます。



ONEPOINT

Indexed Databaseでデータベースを開くには「open()」メソッドを使う

Indexed Databaseでデータベースを開くには、「IndexedDB」オブジェクトの「open()」メソッドを使います。「open()」メソッドの最初のパラメータに、開く(または新規に作成する)データベース名を指定します。2番目のパラメータには、データベースのバージョン番号を指定します。

指定したデータベースが開かれた場合、「success」イベントが発生します。イベントハンドラにはイベントオブジェクトが渡され、「result」プロパティには「resultIDBDatabase」(データベース)オブジェクトが渡されます。「resultIDBDatabase」オブジェクトの「name」プロパティに開いたデータベース名、「version」プロパティにデータベースのバージョンが入ります。

データベースを開くことができなかった場合など、エラーが発生したときには「error」イベントが発生します。

関連項目 ▶▶▶

- Indexed Databaseが使用可能か調べる ……… p.833
- Indexed Databaseでオブジェクトストアを作成する ……… p.837
- Indexed Databaseにデータを書き込む ……… p.841
- Indexed Databaseからデータを読み出す ……… p.845

SECTION-355

Indexed Databaseでオブジェクトストアを作成する

ここでは、実際にデータを格納するオブジェクトストアを作成する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

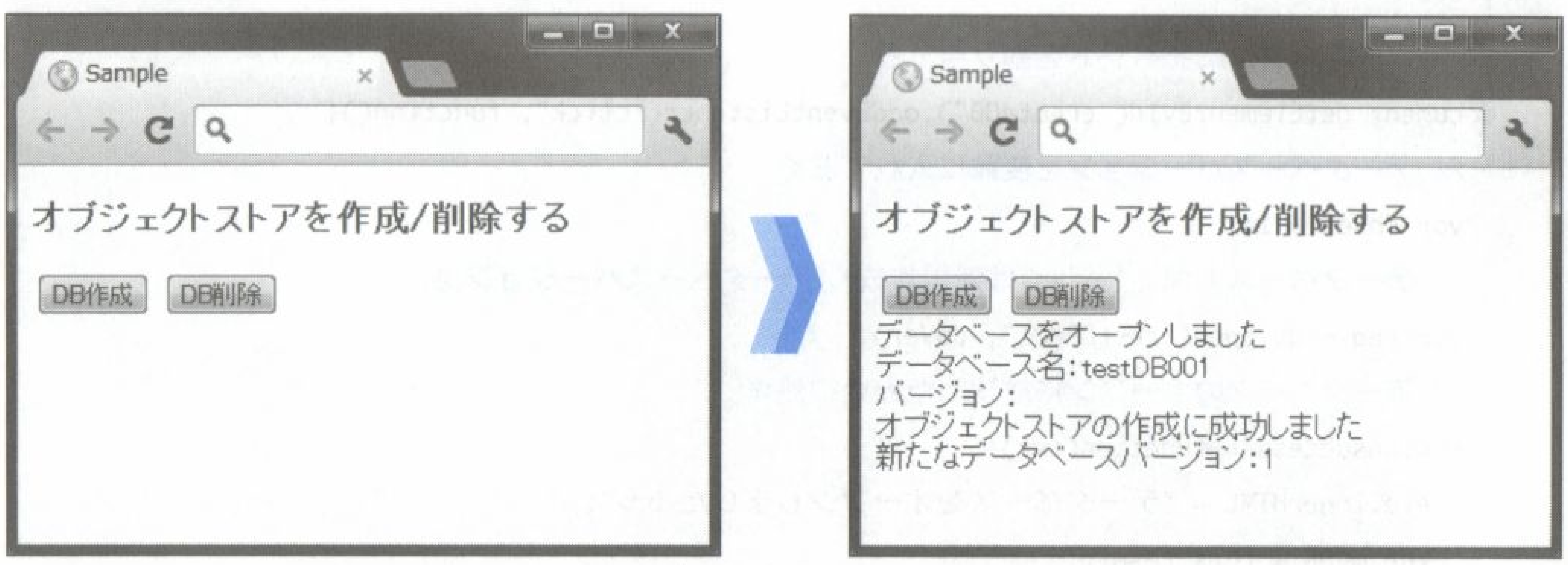
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>オブジェクトストアを作成/削除する</h3>
    <form>
      <input type="button" value="DB作成" id="createDB">
      <input type="button" value="DB削除" id="deleteDB">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Indexed DBオブジェクトが使用できるか調べる
  var db = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB;
  if(!db){
    ele.innerHTML = "Indexed Databaseは使用できません";
    return;
  }
  // DB作成のボタンにイベントを割り当てる
  document.getElementById("createDB").addEventListener("click", function(){
    // データベースバージョンを変数に入れておく
    var dbVer = 1;
    // データベースを開く(もしくは新規作成)。データベースバージョンは1
    var req = db.open("testDB001", dbVer);
    // データベースのオープンに成功したときの処理
    req.onsuccess = function(evt){
      ele.innerHTML = "データベースをオープンしました<br>";
      var myDB = this.result;
      ele.innerHTML += "データベース名:"+myDB.name+"<br>";
      ele.innerHTML += "バージョン:"+myDB.version+"<br>";
      // バージョンチェック
      if (myDB.version != dbVer){
```

```
          // 新たなバージョンを設定
          var setVersion = myDB.setVersion(dbVer);
          // バージョン設定に成功した場合
          setVersion.onsuccess = function(){
            // オブジェクトストアを作成。キーは自動インクリメント
            try{
              var objStore = myDB.createObjectStore("memo");
            }catch(e){
              ele.innerHTML += "すでにオブジェクトストアがあります。削除してから実行してください";
              return;
            }
            ele.innerHTML += "オブジェクトストアの作成に成功しました<br>";
            ele.innerHTML += "新たなデータベースバージョン:"+myDB.version+"<br>";
          }
        }else{
          ele.innerHTML += "オブジェクトストアはすでに作成済みです";
        }
      }
      // データベースのオープンに失敗したときの処理
      req.onerror = function(evt){
        ele.innerHTML = "データベースのオープンに失敗しました";
      }
    }, false);
    // DB削除のボタンにイベントを割り当てる
    document.getElementById("deleteDB").addEventListener("click", function(){
      db.deleteDatabase("testDB001");
      ele.innerHTML += "データベースを削除しました";
    }, false);
  }, false);
```

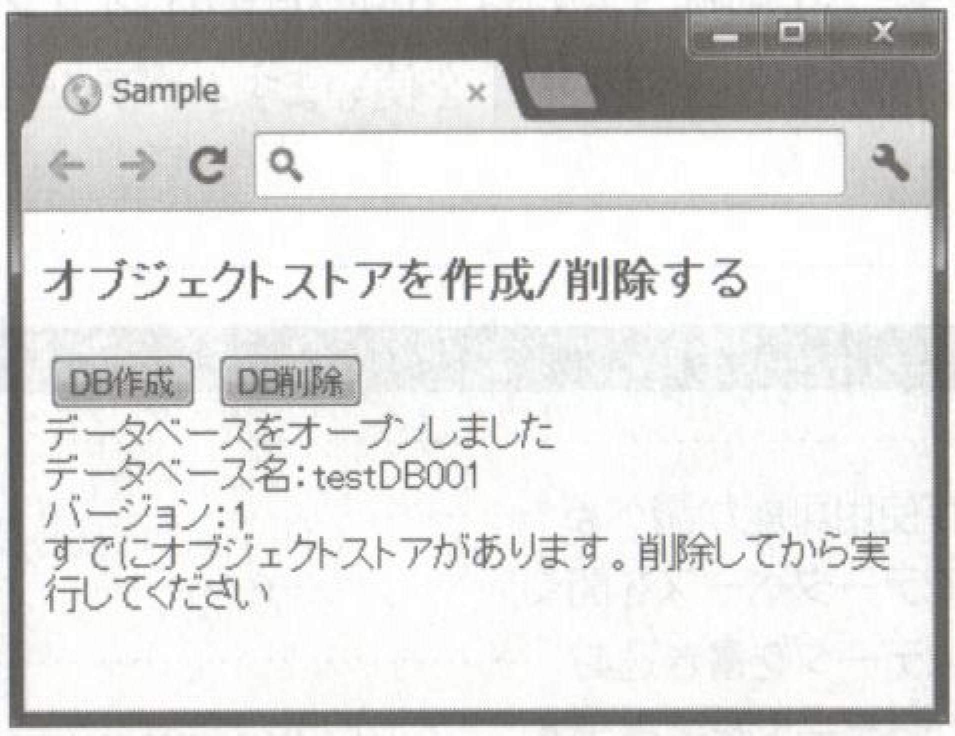
ブラウザでindex.htmlを表示し、「DB作成」ボタンをクリックすると、オブジェクトストアが作成されます(オブジェクトストアが作成されていない初回の場合)。



オブジェクトストア作成後に「DB作成」ボタンをクリックすると、作成済みであると表示されます。



データベースバージョンを変更してから再度、ページを読み込み、「DB作成」ボタンをクリックすると、削除してから実行するように表示されます。



「DB削除」ボタンをクリックすると、データベースが削除されます。



ONEPOINT

Indexed Databaseでオブジェクトストアを作成するには「createObjectStore()」メソッドを使う

実際にデータを格納する領域であるオブジェクトストアを作成するには、データベースが正常に開かれた後に行います。オブジェクトストアを作成する場合、データベースのバージョン番号を変更した後に呼び出されるイベントハンドラ内で行います。イベントハンドラ内では、「createObjectStore()」メソッドを使います。「createObjectStore()」メソッドのパラメータには、オブジェクトストアの名前を指定します。また、2番目以降のパラメータもありますが、省略することができます。2番目のパラメータはオブジェクトリテラル型式で指定します。「keyPath」(キーのパス)、「autoIncrement」(キーの自動インクリメント)を指定することができます。

作成済みのデータベースを削除するには「IndexedDB」オブジェクト(「IDBFactory」オブジェクト)の「deleteDatabase()」メソッドを使います。パラメータには、削除したいデータベース名を指定します。

関連項目 ▶▶▶

- ファイルを作成する ………… p.696
- Indexed Databaseが使用可能か調べる ………… p.833
- Indexed Databaseでデータベースを開く ………… p.835
- Indexed Databaseにデータを書き込む ………… p.841
- Indexed Databaseからデータを読み出す ………… p.845

SECTION-356

Indexed Databaseにデータを書き込む

ここでは、オブジェクトストアにデータを書き込む方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>データを書き込む</h3>
    <form>
      <input type="button" value="DBを開く" id="openDB">
      <input type="button" value="DB削除" id="deleteDB"><br>
      メモ：<input type="text" value="db test" id="memoData">
      <input type="button" value="DBに書き込み" id="saveDB">
    </form>
    <output></output>
  </body>
</html>
```

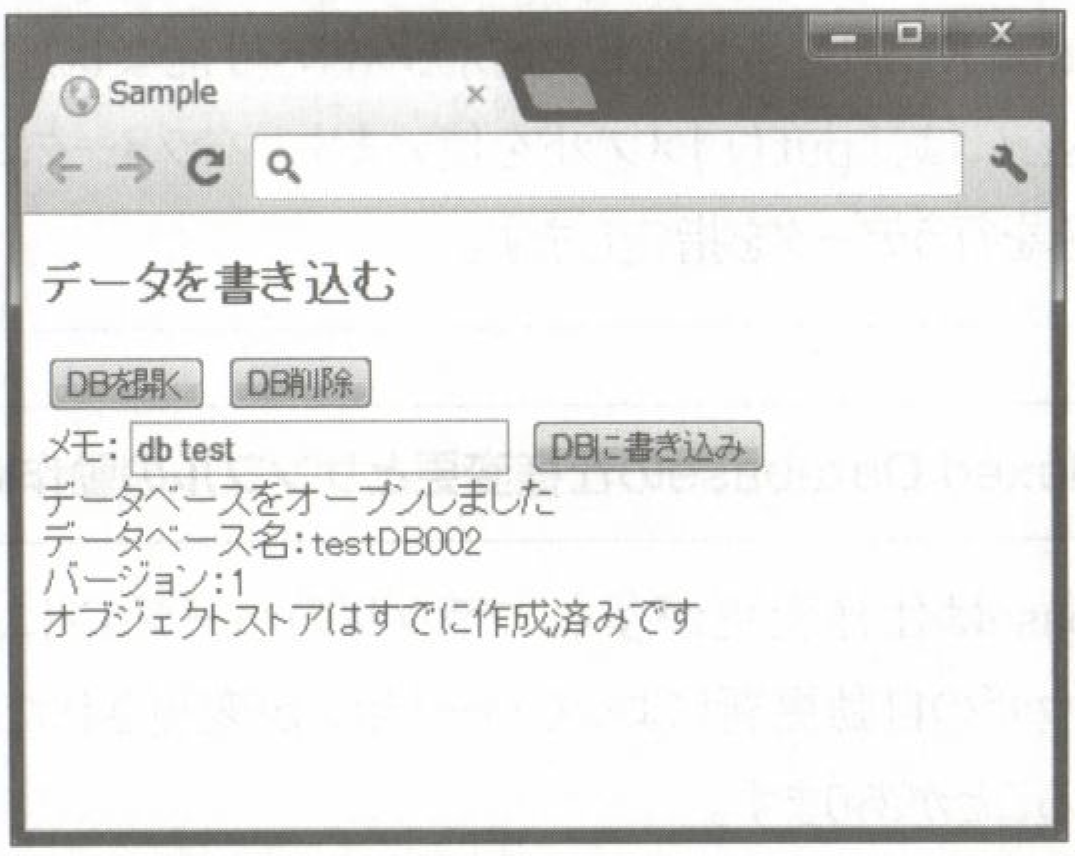
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Indexed DBオブジェクトの読み出し
  var db = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB;
  var IDBTransaction =
    window.IDBTransaction || window.webkitIDBTransaction || window.mozIDBTransaction;
  var IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange || window.mozIDBKeyRange;
  var myDB = null;  // データベースへの有無を示す変数
  // DB作成のボタンにイベントを割り当てる
  document.getElementById("openDB").addEventListener("click", function(){
    // データベースバージョンを変数に入れておく
    var dbVer = 1;
    // データベースを開く(もしくは新規作成)。データベースバージョンは1
    var req = db.open("testDB002", dbVer);
    // データベースのオープンに成功した時の処理
    req.onsuccess = function(evt){
      ele.innerHTML = "データベースをオープンしました<br>";
      myDB = this.result;
      ele.innerHTML += "データベース名："+myDB.name+"<br>";
```

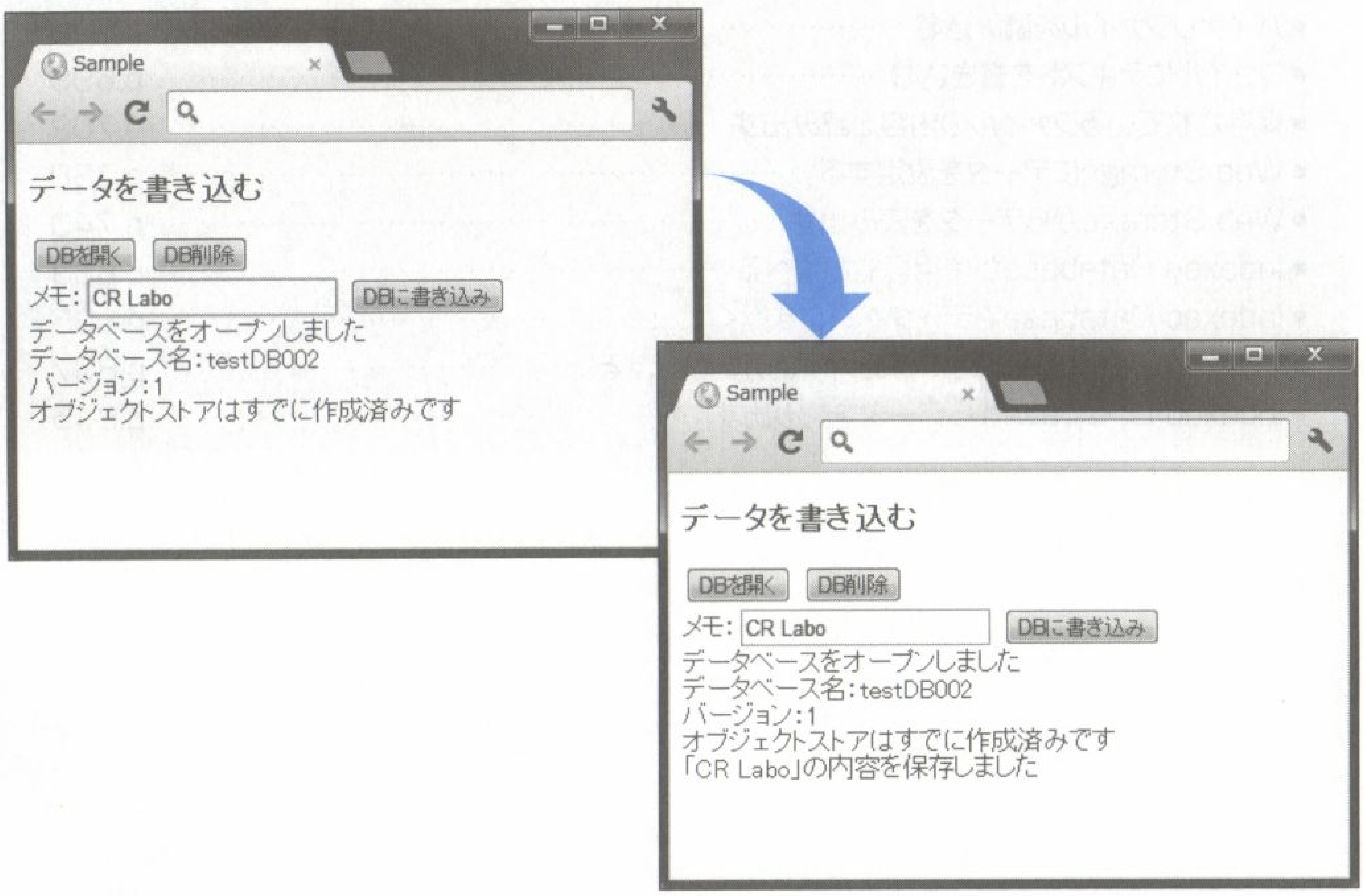
```
      ele.innerHTML += "バージョン:"+myDB.version+"<br>";
      // バージョンチェック
      if (myDB.version != dbVer){
        // 新たなバージョンを設定
        var setVersion = myDB.setVersion(dbVer);
        // バージョン設定に成功した場合
        setVersion.onsuccess = function(){
          // オブジェクトストアを作成。キーは自動インクリメント
          try{
            // Chromeでは{ autoIncrement : true }がないとエラー
            var objStore = myDB.createObjectStore("memo", { autoIncrement : true });
          }catch(e){
            ele.innerHTML += "すでにオブジェクトストアがあります。削除してから実行してください";
            return;
          }
          ele.innerHTML += "オブジェクトストアの作成に成功しました<br>";
          ele.innerHTML += "新たなデータベースバージョン:"+myDB.version+"<br>";
        }
      }else{
        ele.innerHTML += "オブジェクトストアはすでに作成済みです<br>";
      }
    }
    // データベースのオープンに失敗したときの処理
    req.onerror = function(evt){
      ele.innerHTML = "データベースのオープンに失敗しました<br>";
    }
  }, false);
  // DB削除のボタンにイベントを割り当てる
  document.getElementById("deleteDB").addEventListener("click", function(){
    db.deleteDatabase("testDB001");
    ele.innerHTML += "データベースを削除しました<br>";
  }, false);
  // DB保存のボタンにイベントを割り当てる
  document.getElementById("saveDB").addEventListener("click", function(){
    // データベースが開かれているか確認
    if (!myDB){
      ele.innerHTML += "データベースが開かれていません<br>";
      return;
    }
    // テキストフィールドに入力されているメモの内容を読み出す
    var memotext = document.getElementById("memoData").value;
    // トランザクションの設定。memoに対して読み書き(READ_WRITE=1)
    var trans = myDB.transaction(["memo"], IDBTransaction.READ_WRITE);
    var store = trans.objectStore("memo");
    // メモ内容を書き込み
    store.put({ "mymemo" : memotext });
    ele.innerHTML += "「"+memotext+"」の内容を保存しました<br>";
```

```
    }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「DBを開く」ボタンをクリックすると、データベースが開きます。データベースが存在しない場合は、新規に作成されます。



テキストフィールドに文字を入力し、「DBに書き込み」ボタンをクリックすると、データベースに書き込みが行われます。



ONEPOINT **オブジェクトストアにデータを書き込むにはトランザクションを利用する**

オブジェクトストアにデータを書き込むには、トランザクションを利用します。「transaction()」メソッドの最初のパラメータにオブジェクトストア名、2番目のパラメータには読み出しか読み書きなのかを指定します。「IDBTransaction.READ_ONLY」なら読み出しのみ、「IDBTransaction.READ_WRITE」なら読み書き可能な状態で処理されます。実際にデータを書き込むには、「put()」メソッドを使います。パラメータには、オブジェクトリテラル型式で書き込みを行うデータを指定します。

COLUMN **Indexed Databaseの仕様変更とサンプルの動作について**

Indexed Databaseは仕様変更が多く、ブラウザのバージョンによって動作しない場合があります。ブラウザの自動更新によってバージョンが変更された場合、サンプルプログラムが動かなくなることがあります。

関連項目 ▶▶▶

- テキストファイルを読み込む …… p.683
- バイナリファイルを読み込む …… p.685
- ファイルにテキストを書き込む …… p.699
- 保存されているファイルの内容を読み出す …… p.702
- Web Storageにデータを設定する …… p.738
- Web Storageからデータを読み出す …… p.740
- Indexed Databaseが使用可能か調べる …… p.833
- Indexed Databaseでデータベースを開く …… p.835
- Indexed Databaseでオブジェクトストアを作成する …… p.837
- Indexed Databaseからデータを読み出す …… p.845

SECTION-357

Indexed Databaseからデータを読み出す

ここでは、オブジェクトストアに保存されているデータを読み出す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

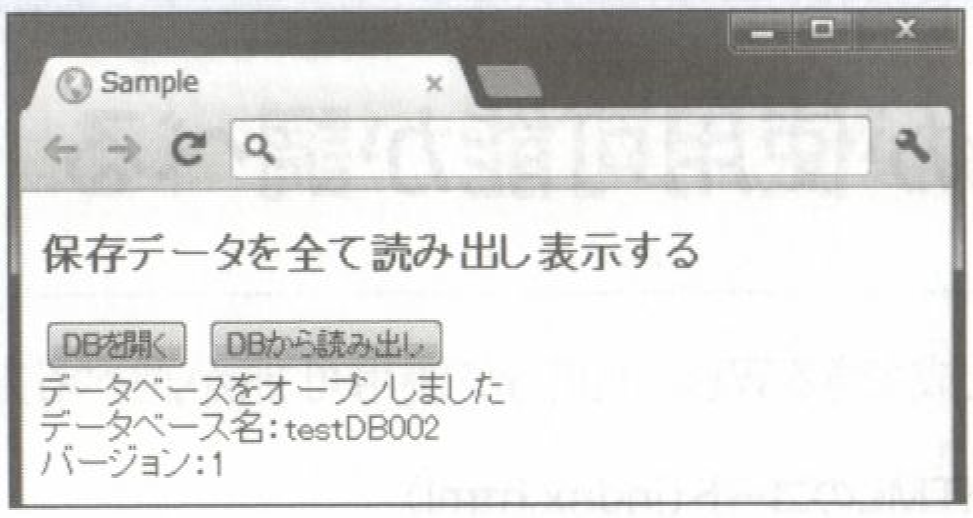
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>保存データを全て読み出し表示する</h3>
    <form>
      <input type="button" value="DBを開く" id="openDB">
      <input type="button" value="DBから読み出し" id="readDB">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

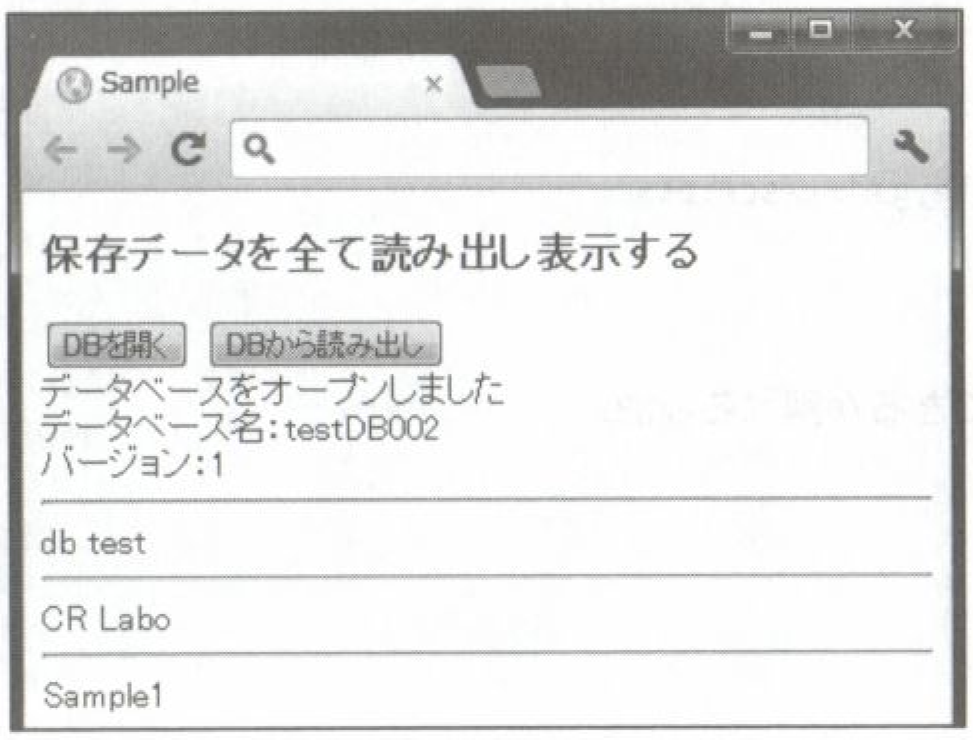
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Indexed DBオブジェクトの読み出し
  var db = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB;
  var IDBTransaction =
    window.IDBTransaction || window.webkitIDBTransaction || window.mozIDBTransaction;
  var IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange || window.mozIDBKeyRange;
  var myDB = null;  // データベースへの有無を示す変数
  // DB作成のボタンにイベントを割り当てる
  document.getElementById("openDB").addEventListener("click", function(){
    // データベースバージョンを変数に入れておく
    var dbVer = 1;
    // データベースを開く(もしくは新規作成)。データベースバージョンは1
    var req = db.open("testDB002", dbVer);
    // データベースのオープンに成功したときの処理
    req.onsuccess = function(evt){
      ele.innerHTML = "データベースをオープンしました<br>";
      myDB = this.result;
      ele.innerHTML += "データベース名:"+myDB.name+"<br>";
      ele.innerHTML += "バージョン:"+myDB.version+"<br>";
    }
```

```
    // データベースのオープンに失敗したときの処理
    req.onerror = function(evt){
      ele.innerHTML = "データベースのオープンに失敗しました<br>";
    }
  }, false);
  // DB読み出しのボタンにイベントを割り当てる
  document.getElementById("readDB").addEventListener("click", function(){
    // データベースが開かれているか確認
    if (!myDB){
      ele.innerHTML += "データベースが開かれていません<br>";
      return;
    }
    // トランザクションの設定。memoに対しては読み出し専用(READ_ONLY=0)
    var trans = myDB.transaction(["memo"], IDBTransaction.READ_ONLY);
    var store = trans.objectStore("memo");
    var req = store.count();
    // 正しく読み出せた場合の処理
    req.onsuccess = function(evt) {
      // 登録合計数が入る
      var total = evt.target.result;
      // 最初のメモ内容を読み出し
      for(var i=1; i<total; i++){
        var req = store.get(i);
        // 正しく読み出せた場合の処理
        req.onsuccess = function(evt) {
          var data = evt.target.result;
          // undefinedならデータがない
          if (!data){
            ele.innerHTML += "<hr>データがありません<br>";
            return;
          }
          var text = data.mymemo;
          // 読み出した結果を表示
          ele.innerHTML += "<hr>"+text+"<br>";
        }
      }
    }
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「DBを開く」ボタンをクリックすると、データベースが開きます。



「DBから読み出し」ボタンをクリックすると、データベース内に書き込まれたデータが表示されます。



ONEPOINT　オブジェクトストアのデータを読み出すにはトランザクションを利用する

オブジェクトストアのデータを読み出すには、トランザクションを利用します。「transaction()」メソッドの最初のパラメータにオブジェクトストア名、2番目のパラメータには読み出しであることを指定します。

データを読み出すには、「get()」メソッドを使います。パラメータには、キー番号を指定します。番号は1から始まり、登録されているデータ数分までになります。登録されているデータの数は、オブジェクトストアの「count()」メソッドで取得することができます。「count()」メソッドも非同期なので、「onsuccess」にコールバック関数を指定します。

COLUMN　Indexed Databaseの仕様変更とサンプルの動作について

Indexed Databaseは仕様変更が多く、ブラウザのバージョンによって動作しない場合があります。ブラウザの自動更新によってバージョンが変更された場合、サンプルプログラムが動かなくなることがあります。

関連項目 ▶▶▶

- Web Storageからデータを読み出す …… p.740
- Indexed Databaseが使用可能か調べる …… p.833
- Indexed Databaseでデータベースを開く …… p.835
- Indexed Databaseでオブジェクトストアを作成する …… p.837
- Indexed Databaseにデータを書き込む …… p.841

Web Audioが使用可能か調べる

ここでは、自由に音を生成できるWeb Audioが使用可能か調べる方法について解説します。

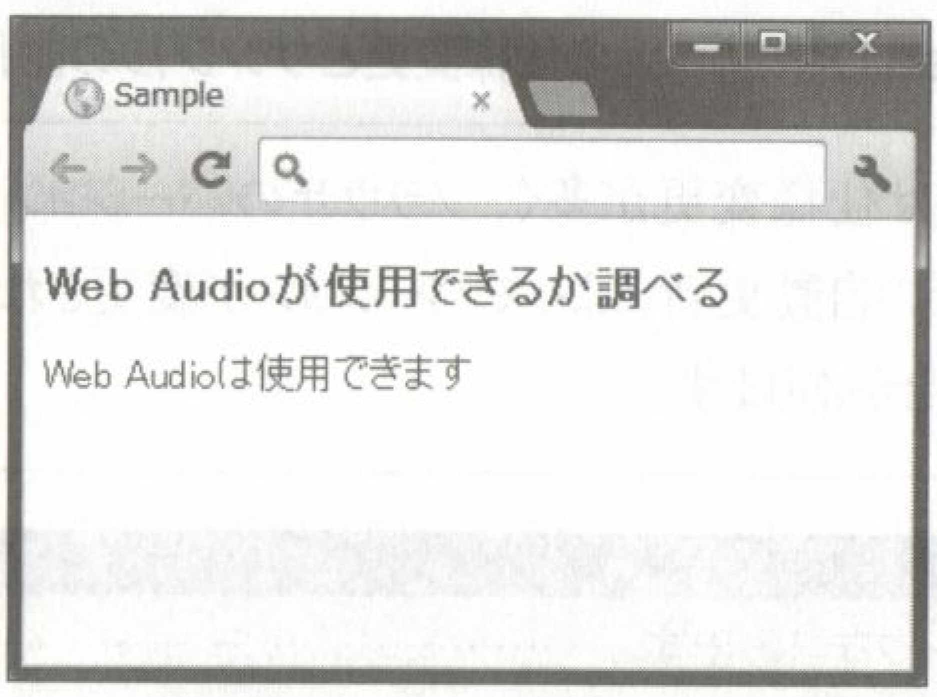
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Audioが使用できるか調べる</h3>
    <output></output>
  </body>
</html>
```

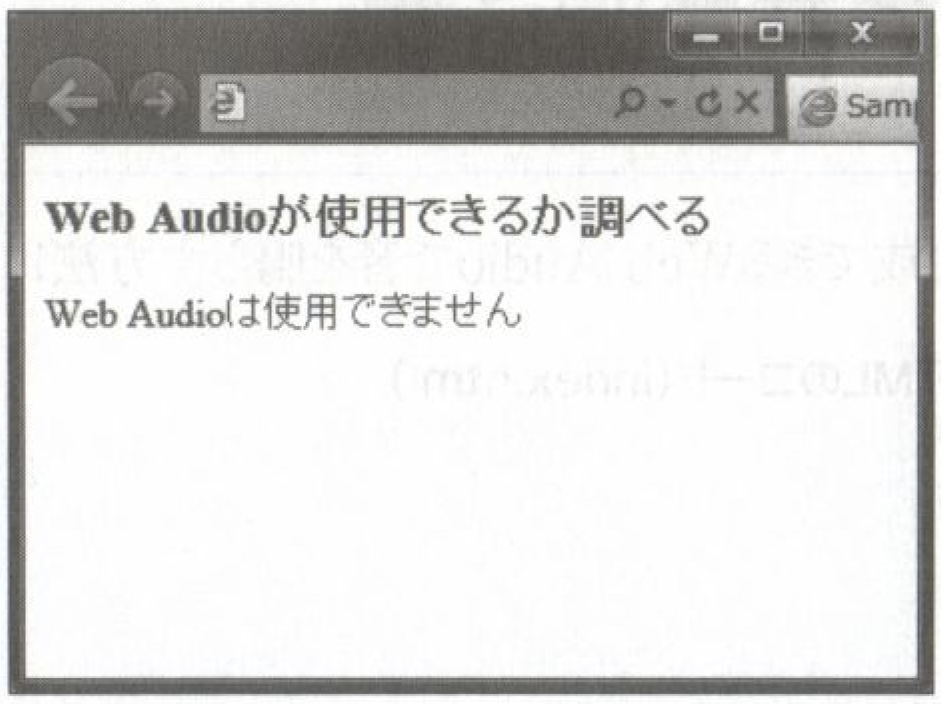
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Audioオブジェクトが使用できるか調べる
  if(window.webkitAudioContext){
    ele.innerHTML = "Web Audioは使用できます";
  }else{
    ele.innerHTML = "Web Audioは使用できません";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、Web Audioが使用できる場合は、次のように表示されます。



Web Audioが使用できない場合は、次のように表示されます。



ONEPOINT Web Audioが利用できるかどうかは「audioContext」オブジェクトの有無を調べる

HTML5では音を扱う場合、「audio」要素(「Audio」オブジェクト)が使われます。ただし、「audio」要素を使用した場合は、あらかじめ用意されているサウンドファイルか、リアルタイムに入力したサウンドのどちらかになります。このため、周波数を指定して任意の音を生成するといった用途には使えません。

任意の音を生成する場合には、Web Audio APIを利用します。Web Audio APIを利用すれば、任意の音を作り出すことができます。

Web Audioが使えるかどうかは、「window」オブジェクト内に「audioContext」オブジェクトがあるかどうか調べます。「audioContext」オブジェクトがない場合は、「undefined」になります。なお、2012年7月時点では、「webkitAudioContext」のようにベンダープレフィックスを付けないと動作しません。

関連項目 ▶▶▶

●オーディオを再生・停止する ………………………… p.610
●Web Audioで音を鳴らす ………………………… p.850
●Web Audioでファイルを読み込んで演奏する ………………………… p.852
●Web Audioでボリュームを指定する ………………………… p.855

SECTION-359

Web Audioで音を鳴らす

ここでは、自由に音を生成できるWeb Audioで音を鳴らす方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Audioで音を鳴らす</h3>
    <form>
      <input type="button" value="Beep" id="playSound">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Audio APIが使えるか調べる
  if (!window.webkitAudioContext){
    ele.innerHTML = "Web Audio APIは使えません";
    return;
  }
  // ボタンにイベントを設定する
  document.getElementById("playSound").addEventListener("click", function(){
    // AudioContextオブジェクトを取得(WebKit)
    var context = new webkitAudioContext();
    // 音を鳴らすためのバッファを生成
    var audioSource = context.createBufferSource();
    // バッファのチャンネル数は1、3秒分の4096バイトバッファ、サンプリングレートは48KHz
    audioSource.buffer = context.createBuffer( 1, 4096*3, 48000 );
    // オーディオソース(ノード)を接続
    audioSource.connect(context.destination);
    // 最初のチャンネルのバッファに直接アクセス
    var data = audioSource.buffer.getChannelData(0);
    // サイン波形を生成してバッファに入れる
    for (var i=0; i<data.length; i++) {
      data[i] = Math.floor(Math.sin(i * Math.PI/180));
```

```
      }
      // 0秒から再生
      audioSource.noteOn(0);
   }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、音が鳴ります。



ONEPOINT Web Audio APIで音を鳴らすにはオーディオコンテキストを生成する

Web Audio APIで音を鳴らすには、オーディオコンテキストを生成します。オーディオコンテキストは、「new webkitAudioContext()」として生成します(ベンダープレフィックス付きになっている)。次に、生成したオーディオコンテキストの「createBufferSource()」メソッドを使って、音を出力するためのオーディオソース(ノード)を生成します。

Web Audio APIは、複数のノードを合成(波形合成)して音を鳴らします。単純に音を鳴らすには、オーディオソースの「buffer」プロパティに、実際に音を出すためのデータを入れます。これは、「createBuffer()」メソッドを使って行います。「createBuffer()」メソッドでは、チャンネル数、バッファサイズ、サンプリングレートの3つを指定します。

あとは、生成したバッファに音を出すための数値を設定します。サンプルではモノラルなので、「getChannelData(0)」としてチャンネルのバッファを取得し、そのバッファに音量を入れています。

バッファに鳴らすデータを用意したら、オーディオソースの「noteOn()」メソッドを使って音を出します。「noteOn()」メソッドのパラメータには、再生を開始までの秒数を指定します。「noteOn(0)」ならすぐに再生し、「noteOn(3)」なら3秒後に再生します。

関連項目 ▶▶▶

- オーディオを再生・停止する ………… p.610
- Web Audioが使用可能か調べる ………… p.848
- Web Audioでファイルを読み込んで演奏する ………… p.852
- Web Audioでボリュームを指定する ………… p.855

SECTION-360

Web Audioでファイルを読み込んで演奏する

ここでは、Web Audioでサーバー上にあるファイルを読み込んで演奏する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Audioでファイルを読み込み演奏する</h3>
    <form>
      <input type="button" value="再生" id="playSound">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Web Audio APIが使えるか調べる
  if (!window.webkitAudioContext){
    ele.innerHTML = "Web Audio APIは使えません";
    return;
  }
  // オーディオファイルのデータを入れておくバッファ
  var buf = null;
  // AudioContextオブジェクトを取得(WebKit)
  var context = new webkitAudioContext();
  // 非同期でオーディオファイルを読み込み
  var xhr = new XMLHttpRequest();
  // sound/sample.mp3ファイルを読み込み
  xhr.open("get", "sound/sample.mp3", true);
  // arraybufferを指定。指定しないとエラーになる
  xhr.responseType = "arraybuffer";
  // データの読み込みが完了した際の処理
  xhr.onload = function(){
    context.decodeAudioData(xhr.response, function(buffer){
```

```
      // 読み込んだデータをバッファに入れる
      buf = buffer;
      ele.innerHTML = "オーディオデータの読み込みが完了しました";
    });
  }
  xhr.send();
  // ボタンにイベントを設定する
  document.getElementById("playSound").addEventListener("click", function(){
    if (!buf){
      ele.innerHTML = "データが完全に読み込まれていません";
      return;
    }
    // バッファを作成
    var audioSource = context.createBufferSource();
    // 読み込んでバッファのデータを設定
    audioSource.buffer = buf;
    // オーディオノードを接続
    audioSource.connect(context.destination);
    // オーディオを再生
    audioSource.noteOn(0);
  }, false);
}, false);
```

HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、ボタンをクリックすると、読み込んだオーディオファイルが演奏されます。



ONEPOINT **Web Audio APIでオーディオファイルを演奏するには非同期通信を使ってファイルを読み込ませておく**

Web Audio APIでオーディオファイルを演奏するには、再生するファイルを非同期通信を使って読み込ませておく必要があります。その際、「responseType」プロパティには、「arraybuffer」のデータ型を指定しておきます。

再生するには、オーディオコンテキストを取得後に、オーディオソースを用意します。これは、「createBufferSource()」メソッドを使って生成し、「buffer」プロパティに非同期通信を使って読み込んだオーディオデータを入れます。その後、オーディオノードを「connect()」メソッドを使って接続します。

あとは、「noteOn()」メソッドを使えば読み込まれたオーディオファイルが再生されます。

関連項目 ▶▶▶

- オーディオを再生・停止する …… p.610
- オーディオデータを読み込む …… p.625
- Web Audioが使用可能か調べる …… p.848
- Web Audioで音を鳴らす …… p.850
- Web Audioでボリュームを指定する …… p.855

SECTION-361

Web Audioでボリュームを指定する

ここでは、Web Audioでボリュームを指定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Audioでボリュームを指定する</h3>
    <form>
      ボリューム：<input type="text" value="0.75" id="volume" size="4">
      <input type="button" value="再生" id="playSound">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

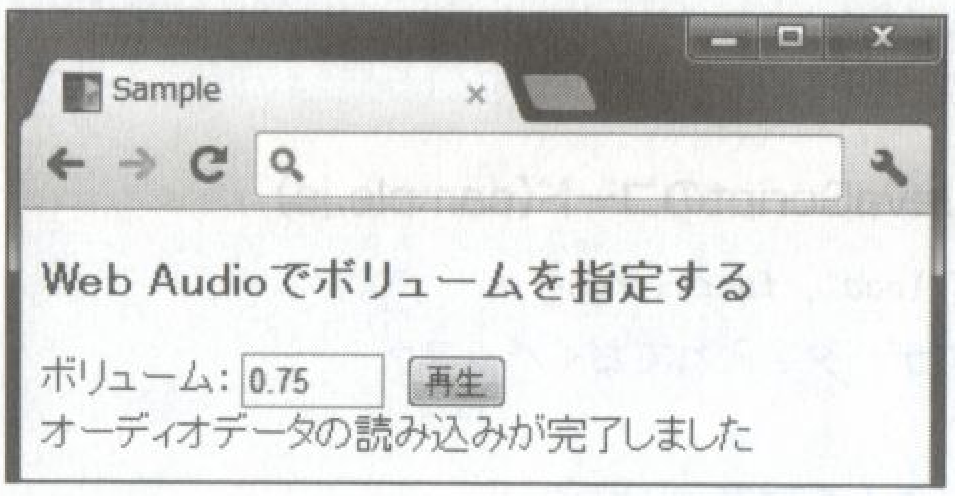
```
window.addEventListener("load", function(){
  // オーディオファイルのデータを入れておくバッファ
  var buf = null;
  // AudioContextオブジェクトを取得(WebKit)
  var context = new webkitAudioContext();
  // 非同期でオーディオファイルを読み込み
  var xhr = new XMLHttpRequest();
  // sound/sample.mp3ファイルを読み込み
  xhr.open("get", "sound/sample.mp3", true);
  // arraybufferを指定。指定しないとエラーになる
  xhr.responseType = "arraybuffer";
  // データの読み込みが完了した際の処理
  xhr.onload = function(){
    context.decodeAudioData(xhr.response, function(buffer){
      // 読み込んだデータをバッファに入れる
      buf = buffer;
      var ele = document.getElementsByTagName("output")[0];
      ele.innerHTML = "オーディオデータの読み込みが完了しました";
    });
  }
  xhr.send();
```

```
    // ボタンにイベントを設定する
    document.getElementById("playSound").addEventListener("click", function(){
      var audioSource = context.createBufferSource(); // バッファを作成
      var gainNode = context.createGainNode(); // ゲインノードを作成
      // 読み込んでバッファのデータを設定
      audioSource.buffer = buf;
      audioSource.connect(gainNode);
      gainNode.connect(context.destination);
      // ボリュームを読み出して設定
      var vol = parseFloat(document.getElementById("volume").value);
      gainNode.gain.value = vol;
      audioSource.noteOn(0); // オーディオを再生
    }, false);
  }, false);
```

HINT

ブラウザによっては、ローカルディスク上では動作しません。その場合は、ネットワーク環境(Webサーバー上)で動作を確認してください。

ブラウザでindex.htmlを表示し、ボリューム(0.0~1.0)を入力した後に「再生」ボタンをクリックすると、読み込んだオーディオファイルが指定したボリュームで演奏されます。



ONEPOINT Web Audio APIでボリュームを指定するには「createGainNode()」メソッドを使ってゲインノードを生成する

Web Audio APIでボリュームを指定するには、「createGainNode()」メソッドを使ってゲインノードを生成します。このゲインノードの「gain.value」プロパティにボリュームを示す小数値を入れます。ボリュームは、「0.0」~「1.0」までの範囲で、値が大きいほどボリュームが大きくなります。

生成したゲインノードをオーディオノードに接続すると、そのオーディオノードの音量が設定されます。あとは、「noteOn()」メソッドを使えば、読み込まれた指定したボリュームで再生されます。

関連項目 ▶▶▶

- オーディオのボリュームを設定・ミュートにする ……… p.620
- Web Audioが使用可能か調べる ……… p.848
- Web Audioで音を鳴らす ……… p.850
- Web Audioでファイルを読み込んで演奏する ……… p.852

SECTION-362

Pointer Lock APIが使用可能か調べる

ここでは、フルスクリーン表示時にマウスポインタがロック可能かどうか調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Pointer Lockサンプル</h3>
    <output></output>
  </body>
</html>
```

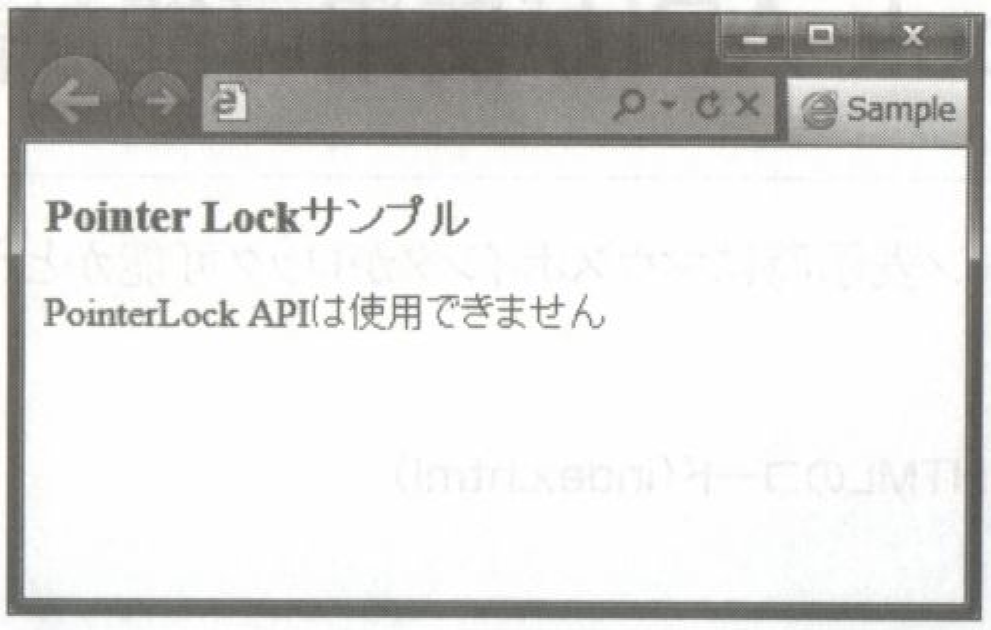
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // PointerLock APIが使用できるか調べる
  var h3ele = document.getElementsByTagName("h3")[0];
  var pLock = h3ele.mozRequestPointerLock;
  if(pLock){
    ele.innerHTML = "PointerLock APIは使用できます";
  }else{
    ele.innerHTML = "PointerLock APIは使用できません";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、Pointer Lock APIが使用できる場合は、次のように表示されます。



Pointer Lock APIが使用できない場合は、次のように表示されます。



ONEPOINT

マウスポインタがロック可能かどうか調べるには「requestPointerLock()」メソッドの有無を調べる

Pointer Lock APIを使うと、フルスクリーンモードでマウスポインタをロックして表示しないようにすることができます。マウスポインタをロックする対象となる要素には「requestPointerLock()」メソッドが用意されています。Pointer Lock APIが使用可能かどうかは、ページ上にある要素に「requestPointerLock()」メソッドがあるかどうかで調べることができます。

なお、2012年7月時点ではFirefoxの場合、メソッド名は「mozRequestPointerLock()」となっています。

SECTION-363

タブを切り換えた場合に処理を行う

ここでは、タブを切り換えた場合に処理を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Page Visibility APIサンプル</h3>
    <output></output>
  </body>
</html>
```

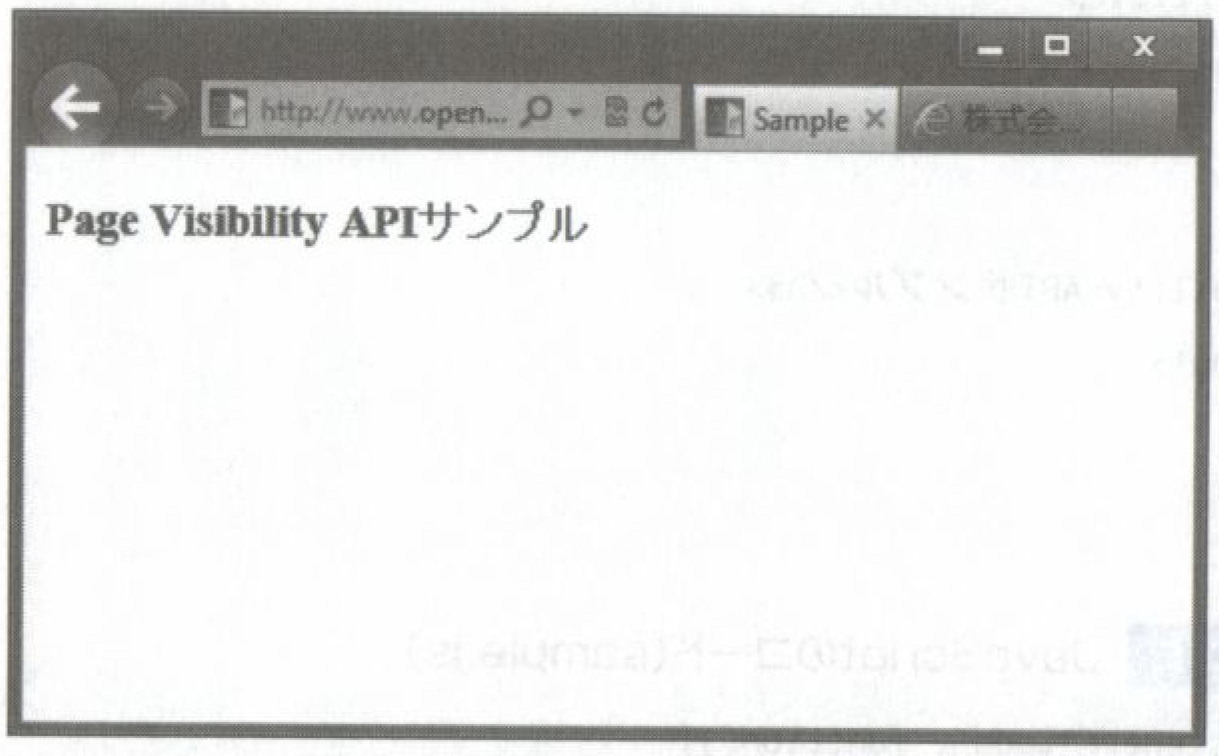
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Google Chromeの場合
  document.addEventListener("webkitvisibilitychange", function(){
    // 状態を表示
    visCheck(document.webkitHidden, document.webkitVisibilityState);
  }, false);
  // Firefoxの場合
  document.addEventListener("mozvisibilitychange", function(){
    // 状態を表示
    visCheck(document.mozHidden, document.mozVisibilityState);
  }, false);
  // IE10の場合
  document.addEventListener("msvisibilitychange", function(){
    // 状態を表示
    visCheck(document.msHidden, document.msVisibilityState);
  }, false);
  // 画面に表示
  function visCheck(visType, visState){
    // 切り替わった時間を取得
    var dateObj = new Date();
    // 表示状態を調べる
    if (visType){
      // ページが隠れた場合
      ele.innerHTML += "ページが隠れました："+dateObj+"<br>";
```

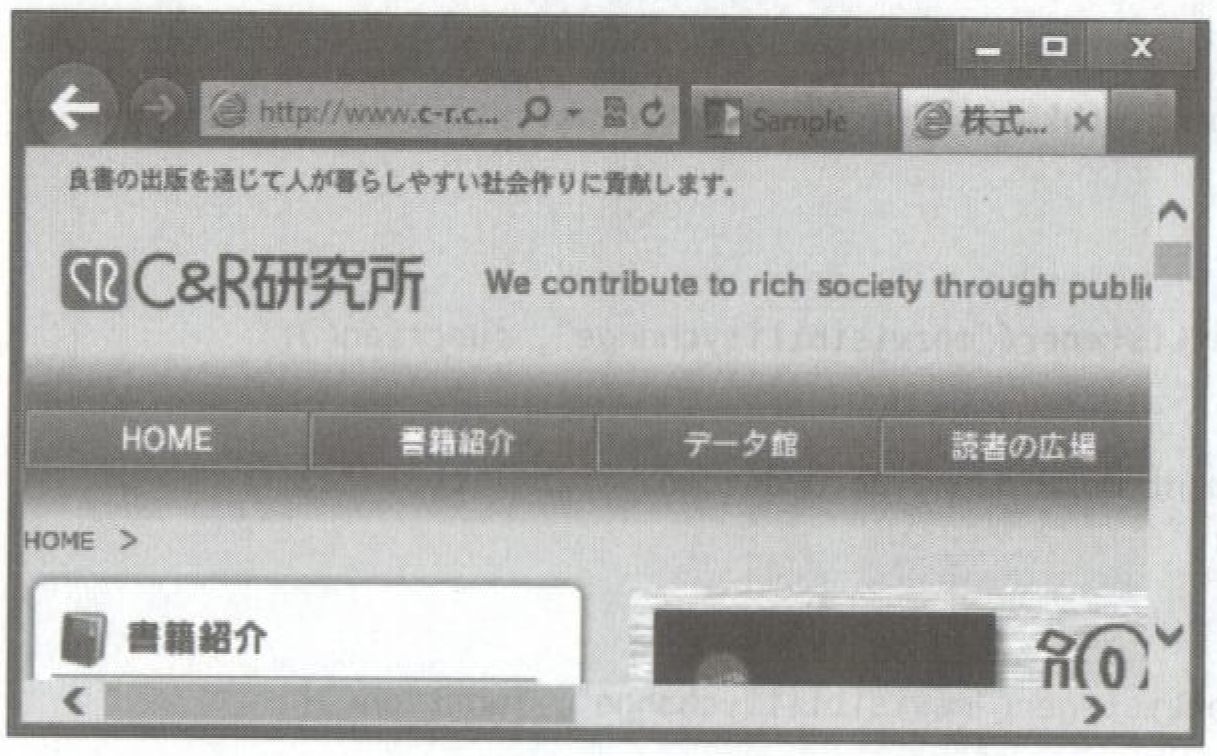
▼

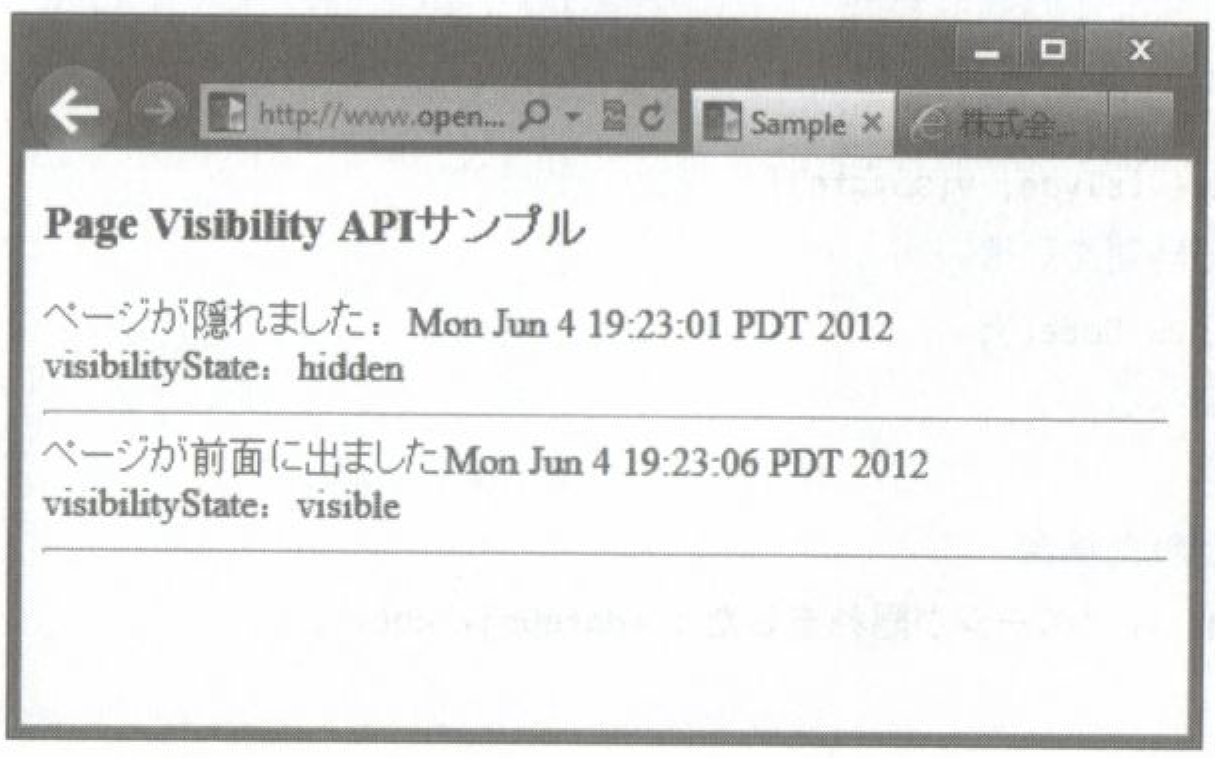
```
    }else{
      // ページが表示された場合
      ele.innerHTML += "ページが前面に出ました"+dateObj+"<br>";
    }
    // ページの状態を表示
    ele.innerHTML += "visibilityState:"+visState+"<hr>";
  }
}, false);
```

ブラウザでindex.htmlを表示すると、ページが前面に出ている状態で表示されます。ここでタブを切り換えます。



他のタブを表示し、再度、タブを切り換えると、いつ切り換えたかを示す時間、および、状態が表示されます。





ONEPOINT **タブが切り替わったときに処理を行うには「visibilitychange」イベントを使う**

Page Visibility APIを使うと、タブが切り替わった際のイベントを取得して処理を行うことができます。タブが切り替わると、「visibilitychange」イベントが発生します。なお、2012年7月時点ではベンダープレフィックスを付けないとイベントが発生しません。

ページが表示されているかどうかは、「document.hidden」プロパティで確認できます。「true」なら非表示、「false」なら表示されていることになります。また、「document.visibilityState」プロパティでも確認できます。「document.visibilityState」プロパティの場合は、より細かい状態を知ることができます。「document.visibilityState」プロパティは、次の表に示す値になります。

値	説明
"hidden"	ページ非表示
"visible"	ページ表示
"preview"	プレビュー
"prerender"	レンダリング中

なお、ウィンドウが切り替わった際に、他のアプリケーションに切り替わった場合は、「visibilitychange」イベントは発生しません。

関連項目 ▶▶▶

- タイマーを設定・解除する …… p.283
- インターバルタイマーを設定・解除する …… p.286
- アニメーション同期タイマーを設定・解除する …… p.289

SECTION-364

要素をドラッグできるようにする

ここでは、要素をドラッグできるようにする方法について解説します。

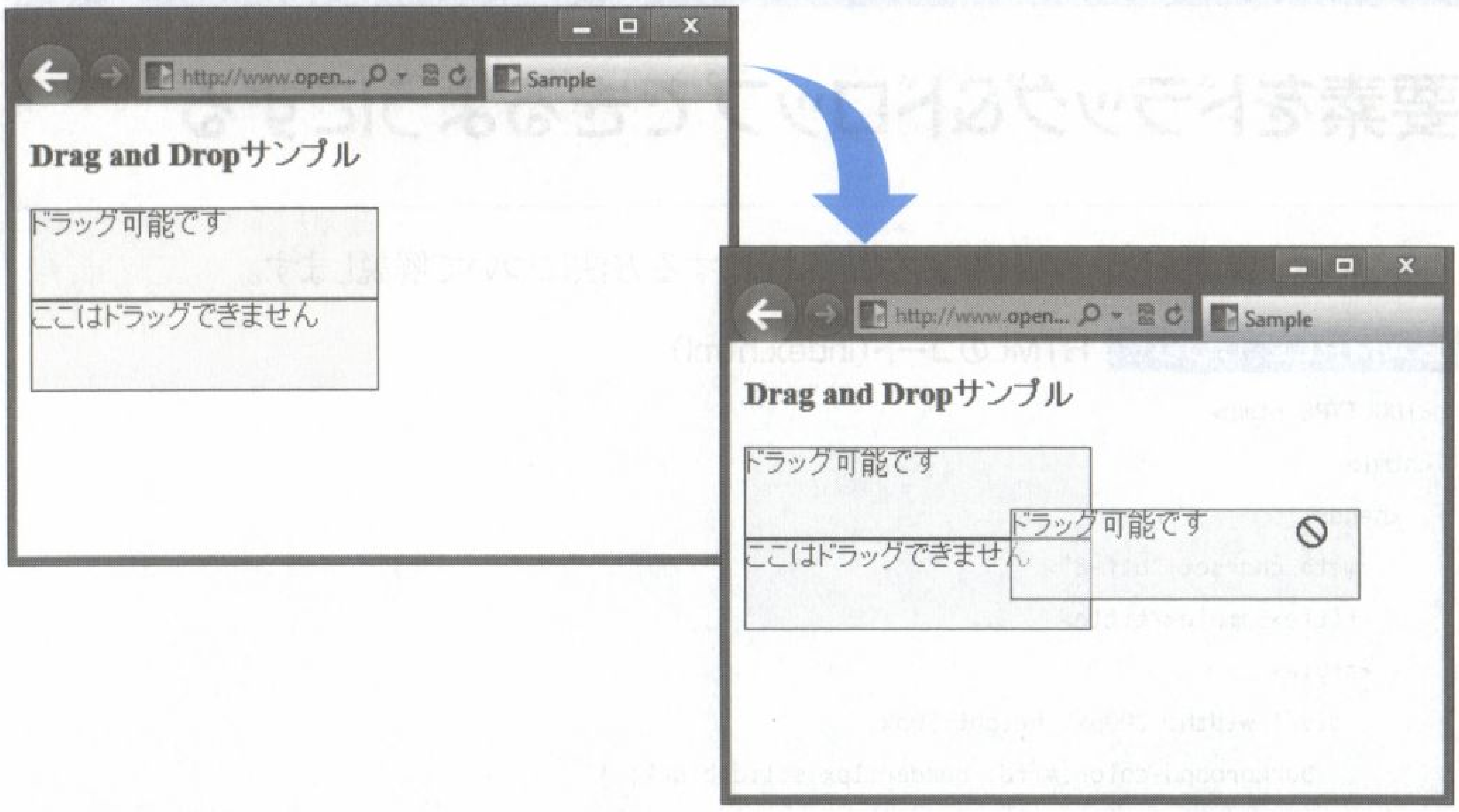
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { width: 200px; height:50px;
        background-color:#ffd; border:1px solid black; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Drag and Dropサンプル</h3>
    <div>ドラッグ可能です</div>
    <div>ここはドラッグできません</div>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var div = document.getElementsByTagName("div")[0];
  // Google Chrome, Safari, IE10の場合
  div.draggable = true;
  // Firefoxの場合は以下が必要
  div.addEventListener("dragstart", function(event){
    event.dataTransfer.setData("text/plain", "dragItem");
  }, true);
}, false);
```

ブラウザでindex.htmlを表示すると、最初の黄色い背景の「div」要素の領域をドラッグすることができます。



ONEPOINT　要素をドラッグできるようにするには「draggable」プロパティを使う

「div」要素など、通常ドラッグできない要素をドラッグできるようにするには、要素の「draggable」プロパティに「true」を指定します。スクリプトでなく、HTML要素に「draggable」属性を指定してもドラッグ可能になります。ただし、Firefoxに関しては、「dragstart」イベントが発生したら、「setData()」メソッドを使ってデータを設定する必要があります。

「draggable」プロパティに「false」を設定すると、その要素はドラッグできなくなります。なお、「img」要素や「a」要素のリンク、テキストなど、デフォルトでドラッグ可能になっている場合は、「draggable」プロパティに「false」を指定しても効果はありません。

関連項目 ▶▶▶

● 要素をドラッグ&ドロップできるようにする ……………… p.864
● ドラッグ要素からドロップ要素にデータを渡す ……………… p.866

SECTION-365

要素をドラッグ&ドロップできるようにする

ここでは、要素をドラッグ&ドロップできるようにする方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

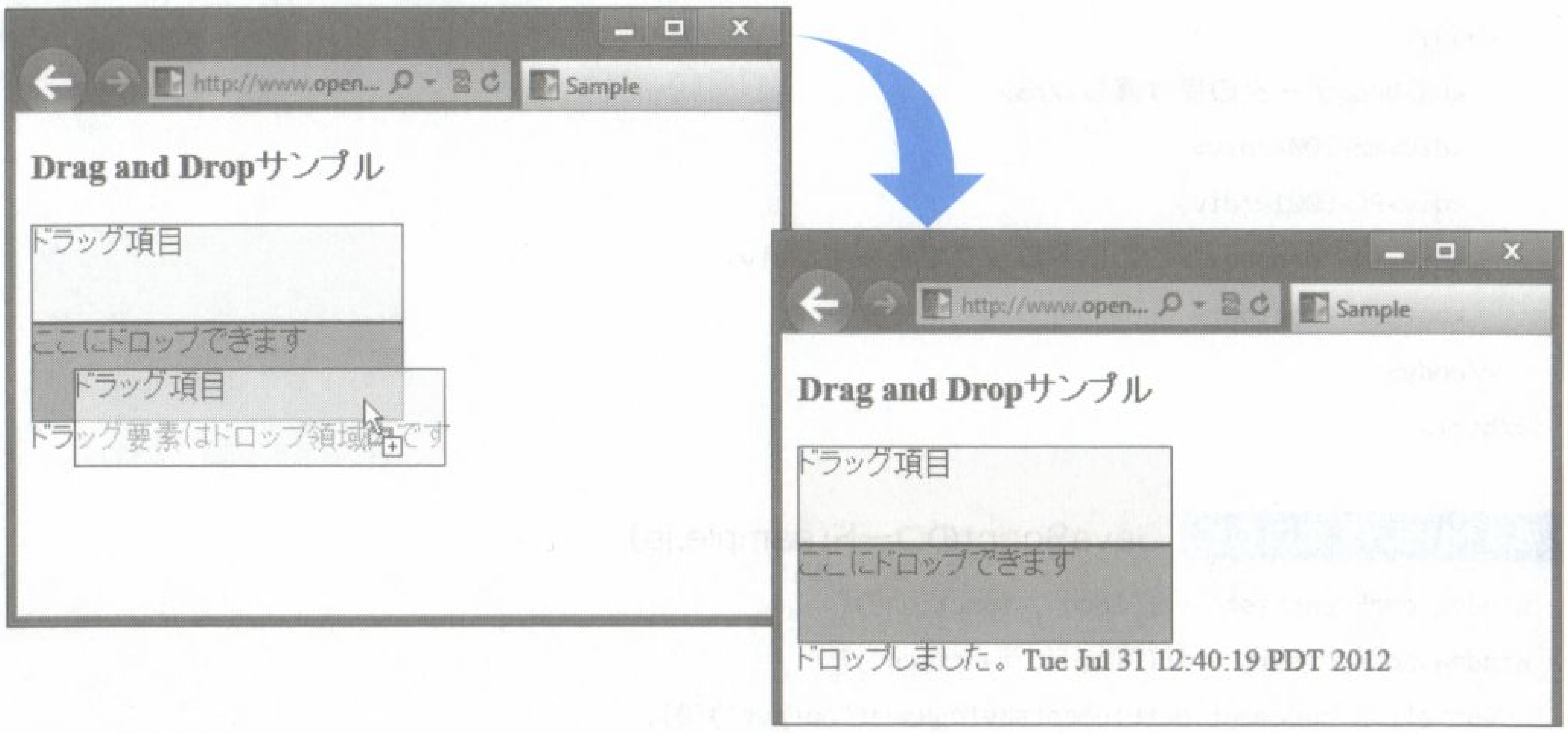
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { width: 200px; height:50px;
        background-color:#ffd; border:1px solid black; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Drag and Dropサンプル</h3>
    <div>ドラッグ項目</div>
    <div>ここにドロップできます</div>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var div = document.getElementsByTagName("div")[0];
  var dropArea = document.getElementsByTagName("div")[1];
  // Google Chrome, Safari, IE10の場合
  div.draggable = true;
  // Firefoxの場合は以下が必要
  div.addEventListener("dragstart", function(event){
    event.dataTransfer.setData("text/plain", "dragItem");
  }, true);
  // ドロップ領域に入った場合の処理
  dropArea.addEventListener("dragover", function(event){
    // 背景色を変える
    this.style.backgroundColor = "orange";
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
    ele.innerHTML = "ドラッグ要素はドロップ領域内です";
  }, true);
  // ドロップ領域から離れた場合の処理
  dropArea.addEventListener("dragleave", function(event){
```

```
    // 背景色を変える
    this.style.backgroundColor = "#ffd";
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
    ele.innerHTML = "ドロップ領域から離れました";
  }, true);
  // ドロップした場合の処理
  dropArea.addEventListener("drop", function(event){
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
    ele.innerHTML = "ドロップしました。"+(new Date());
  }, true);
}, false);
```

ブラウザでindex.htmlを表示し、最初の黄色い背景の「div」要素の領域をドラッグして、下の「div」要素にドロップ領域に重ねると背景色が変化し、ドロップすると、ドロップした日時が表示されます(日時の表示形式はブラウザによって異なる)。



ONEPOINT **ドロップ側の処理は「dragover」「dragleave」「drop」イベントを使う**

「draggable」プロパティに「true」を指定すれば要素をドラッグすることはできますが、ドロップ側の処理は、次のイベントに対して処理を行う必要があります。「dragleave」イベントは省略しても動作します。

イベント	説明
dragover	要素がドロップ領域に重なった(必須)
dragleave	要素がドロップ領域から離れた
drop	要素がドロップされた(必須)

関連項目 ▶▶▶

●要素をドラッグできるようにする……………………………………………………p.862
●ドラッグ要素からドロップ要素にデータを渡す ……………………………………p.866

SECTION-366

ドラッグ要素からドロップ要素にデータを渡す

ここでは、ドラッグ要素からドロップ要素にデータを渡す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

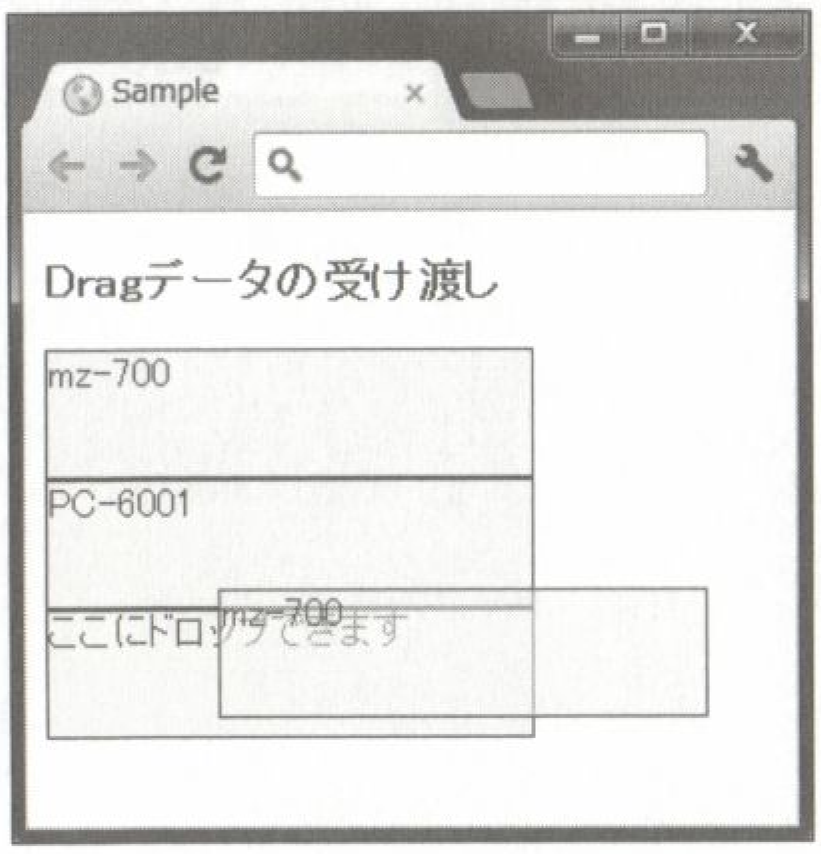
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { width: 200px; height:50px;
        background-color:#ffd; border:1px solid black; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Dragデータの受け渡し</h3>
    <div>mz-700</div>
    <div>PC-6001</div>
    <div id="dropbox">ここにドロップできます</div>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  var div = document.getElementsByTagName("div");
  // ドロップエリアを読み出し
  var dropArea = document.getElementById("dropbox");
  // ドラッグするdiv要素の数だけイベントを設定
  for(var i=0; i<2; i++){
    // div要素をドラッグ可能にする
    div[i].draggable = true;
    // ドラッグ開始時の処理を設定
    div[i].addEventListener("dragstart", function(event){
      // ドロップ先に渡すデータを設定。divの内容を渡す
      event.dataTransfer.setData("text/plain", this.innerHTML);
    }, true);
  }
  // ドロップ領域に入った場合の処理
  dropArea.addEventListener("dragover", function(event){
```

```
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
  }, true);
  // ドロップ領域から離れた場合の処理
  dropArea.addEventListener("dragleave", function(event){
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
  }, true);
  // ドロップした場合の処理
  dropArea.addEventListener("drop", function(event){
    // デフォルトのイベント動作を禁止する
    event.preventDefault();
    // 渡されたデータを読み出す
    var text = event.dataTransfer.getData("text/plain");
    ele.innerHTML = "ドロップ完了。渡されたデータ :"+text;
  }, true);
}, false);
```

ブラウザでindex.htmlを表示し、1番目か2番目の「div」要素の領域をドラッグして一番下の「div」要素にドロップすると、ドロップ先には「div」要素内に表示されているテキストデータが渡され、内容が表示されます。





ONEPOINT データを受け渡すには「setData()」メソッドを使う

ドラッグ先の要素からドロップ先の要素にデータを渡すには、「setData()」メソッドを使います。「setData()」メソッドの最初のパラメータにMIMEタイプ、2番目に渡したいデータを指定します。2番目のデータは、最初のパラメータに指定したMIMEタイプに対応したものになります。

ドロップ先では、「getData()」メソッドを使ってデータを受け取ることができます。「getData()」メソッドのパラメータには、受け取りたいデータのMIMEタイプを指定します。標準テキストであれば、「text/plain」を指定します。

サンプルでは、ページ内にある要素のデータだけでなく、他のアプリケーションからのデータもドロップすることができます。その際、アプリケーションから渡されたデータが表示されます。

関連項目 ▶▶▶

- ファイル名や種類を取得する …… p.680
- テキストファイルを読み込む …… p.683
- バイナリファイルを読み込む …… p.685
- 画像や映像ファイルを読み込む …… p.687
- 要素をドラッグできるようにする …… p.862
- 要素をドラッグ&ドロップできるようにする …… p.864

SECTION-367

通知可能かどうか調べる

ここでは、Web Notifications APIを利用して通知可能か調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Notificationsサンプル</h3>
    <output></output>
  </body>
</html>
```

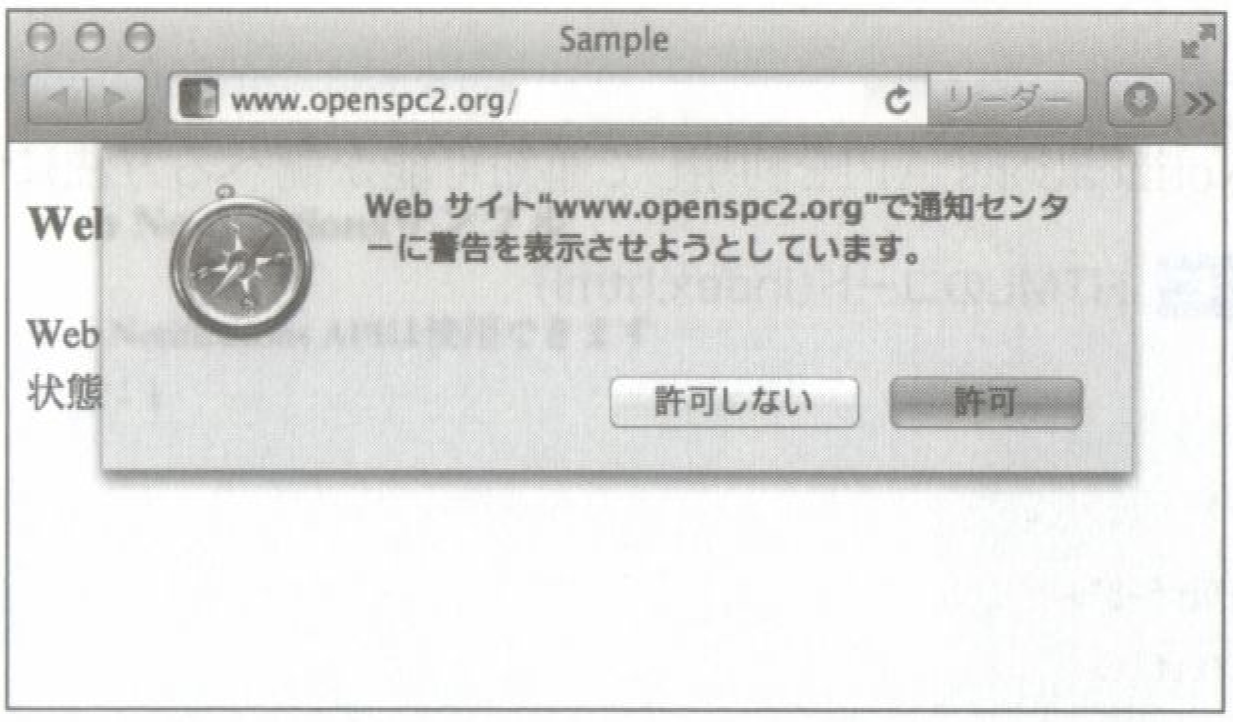
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Google Chrome, Safari 6の場合
  if (webkitNotifications){
    ele.innerHTML = "Web Notifications APIは使用できます<br>";
    // 通知状態を取得
    var stat = webkitNotifications.checkPermission();
    // 通知状況を表示
    ele.innerHTML += "状態 :"+stat+"<br>";
    // 通知が許可されているか調べる
    if(stat == 1){
      // 許可するように要求する
      webkitNotifications.requestPermission(function(){
        // 通知状態を取得して画面に表示
        var stat = webkitNotifications.checkPermission();
        ele.innerHTML += "現在の状態 :"+stat;
      });
    }
  }else{
    ele.innerHTML = "Web Notifications APIは使用できません";
  }
}, false);
```

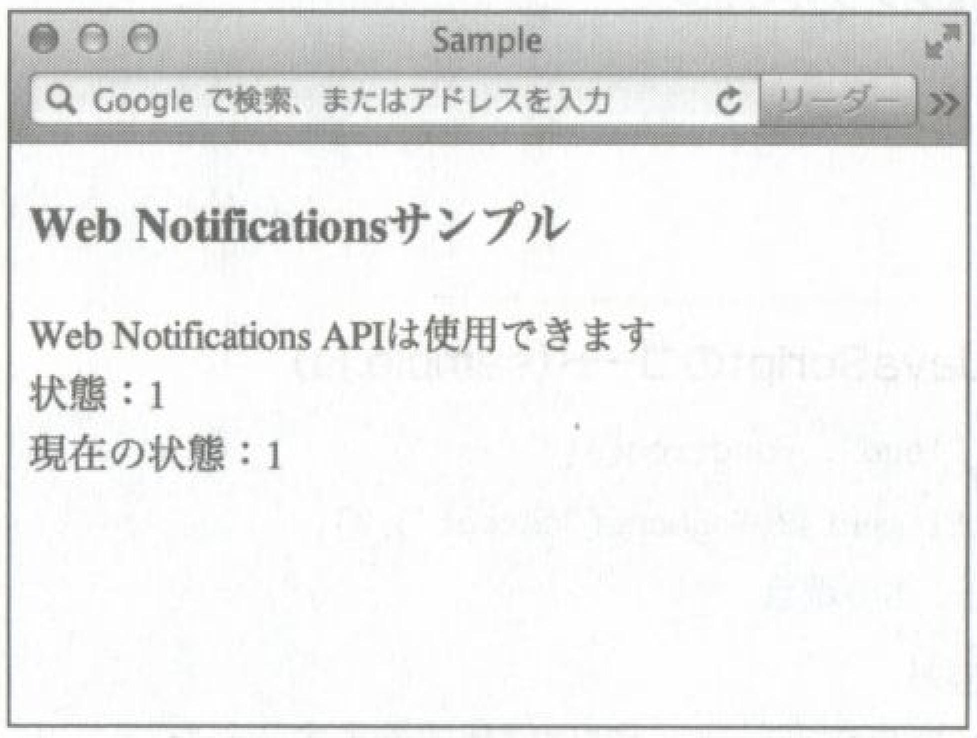
HINT

サンプルの動作については、871ページのCOLUMNを参照してください。

ブラウザでindex.hetmlを表示すると、Web Nofitication APIを利用して処理を行う場合、設定やブラウザによっては確認のダイアログが表示されます。



「許可」ボタンと「許可しない」ボタンのどちらを選択してもコールバック関数が呼び出され、Web Notificationsの状態が表示されます。



ONEPOINT

通知が利用できるかどうかは「webkitNotifications」オブジェクトの有無で調べる

画面の隅にメッセージを表示する（通知する）には、Web Notifications APIを利用します。Web Notifications APIが利用できるかどうかは、「webkitNotifications」オブジェクトの有無で調べることができます。オブジェクトがあれば使用可能、なければ使用することはできません。

また、ユーザーが通知を許可していない場合は、使用することができません。ユーザーが許可しているかどうかは、「checkPermission()」メソッドで調べることができます。返される数値と内容は、次の表のようになります。なお、Safari 6では通知を許可している場合でも「1」を返すことがあります。

値	キーワード	説明
0	"default"	デフォルト設定
1	"denied"	通知を許可していない
2	"granted"	通知は許可されている

通知を許可するように要求する場合は、「requestPermission()」メソッドを使います。許可/不許可に関係なく、パラメーターに指定したコールバック関数が呼び出されます。

COLUMN Safari 6での動作について

Safari 6の場合、Web Notifications APIは、OS X 10.8(Mountain Lion)から追加された通知センターの機能を使っています。このため、OS X 10.7(Lion)のSafari 6では動作しません。869〜878ページのサンプルの動作をSafari 6で確認する場合は、OS X 10.8(Mountain Lion)上で確認してください。

COLUMN Google Chromeでの動作について

Google Chromeのデフォルトでは、サイトが通知を表示しようとしたときにページの上部に確認メッセージが表示され、ユーザーが「許可」ボタンをクリックすると、通知を利用できるようになります。ただし、2012年7月現在、通知を利用するサイトにアクセスしただけでは確認メッセージが表示されません(ユーザーのクリックなどが必要)。そのため、869〜878ページのサンプルの動作を確認するには、通知の表示を許可するように設定を変更する必要があります。設定を変更するには、次のように操作します。

❶ ツールバーのレンチアイコンをクリックして[設定]を選択します。
❷ ページ下部の「詳細設定を表示」をクリックします。
❸ 「プライバシー」の[コンテンツの設定]ボタンをクリックします。
❹ 「通知」で[すべてのサイトにデスクトップ通知の表示を許可する]をONにして、[OK]ボタンをクリックします。

なお、サンプルの動作確認が終わったら、セキュリティのため、元の設定に戻すようにしてください。また、ローカルディスク上では正しく動作しないため、ネットワーク環境(Webサーバー上)で確認するようにしてください。

関連項目 ▶▶▶

● 通知する ……………………………… p.872
● 通知を消す ……………………………… p.874
● 通知に関するイベントを処理する ……………… p.876

SECTION-368

通知する

ここでは、Web Notifications APIを利用してデスクトップ上に通知する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Notificationsサンプル</h3>
    <p>アクセス時の時間を通知します</p>
  </body>
</html>
```

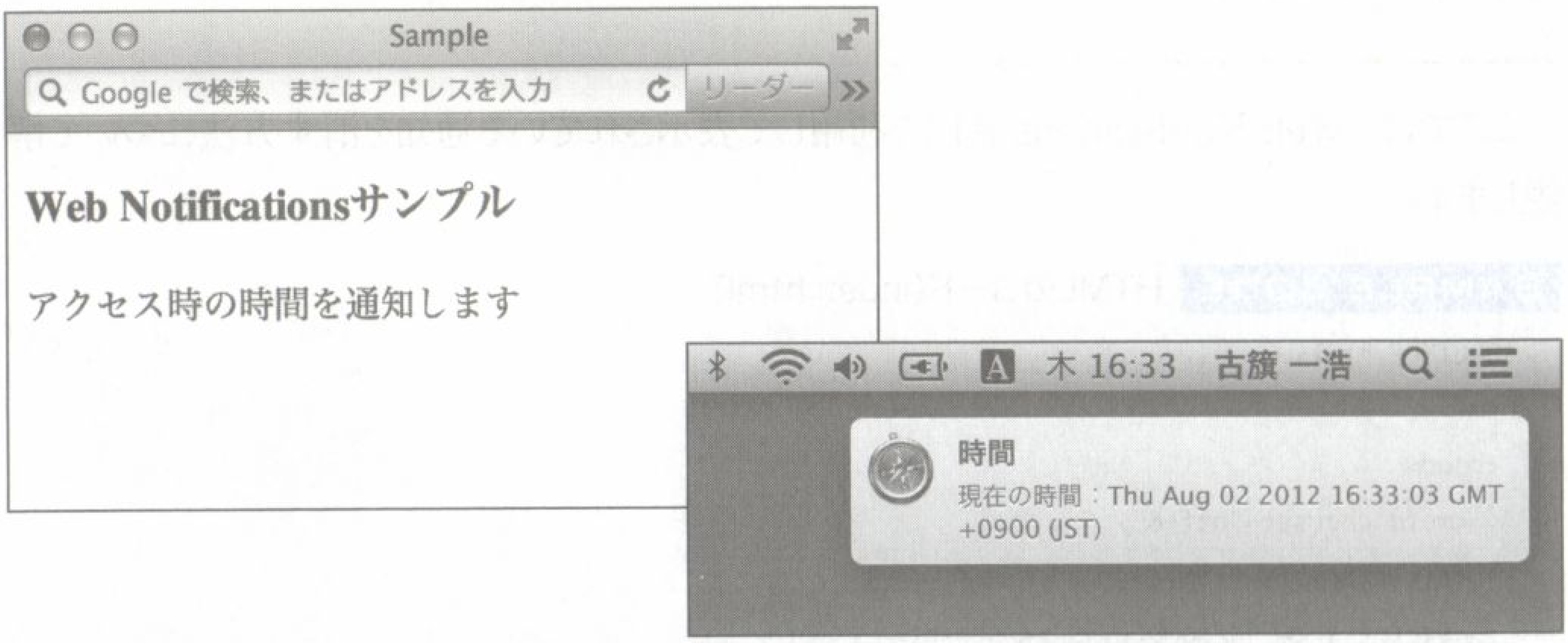
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Google Chrome, Safari 6で使用可能な場合に処理
  if (webkitNotifications){
    // 通知するメッセージを準備
    var message = "現在の時間 :"+(new Date());
    // 通知を行う
    var popup = webkitNotifications.createNotification("icon.png", "時間", message);
    // 画面に表示
    popup.show();
  }
}, false);
```

HINT

Safari 6の場合、OS X 10.8(Mountain Lion)上でのみ動作します。また、Google Chromeで動作を確認する場合は、設定を変更する必要があります。それぞれ、871ページのCOLUMNを参照してください。

ブラウザでindex.htmlを表示し、ページが読み込まれると、その時点でのアクセス時間が通知されます。Macの場合は、画面右上に通知が表示されます。



Windows 7の場合は、画面右下に通知が表示されます。



ONEPOINT 通知するには「createNotification()」メソッドを使う

画面の隅にメッセージを表示する（通知する）には、「createNotification()」メソッドを使います。パラメータは3つあり、最初がアイコンのURL、2番目がタイトル、3番目が実際に表示するメッセージになります。

「createNotification()」メソッドは、生成した「Notification」オブジェクトを返します。この段階では、通知は表示されません。デスクトップに通知を表示するには、「Notification」オブジェクトの「show()」メソッドを実行します。これにより、通知が表示されます。

なお、Safari 6ではアイコンのURLを指定しても反映されず、Safari 6のアプリケーションアイコンが表示されます。

関連項目 ▶▶▶
● アラートダイアログ/確認ダイアログ/入力ダイアログを表示する ……… p.278
● モーダルダイアログを表示する ……… p.281
● 通知可能かどうか調べる ……… p.869
● 通知を消す ……… p.874
● 通知に関するイベントを処理する ……… p.876

SECTION-369

通知を消す

ここでは、Web Notifications APIを利用して表示されている通知を消す方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Notificationsサンプル</h3>
    <p>アクセス時の時間を通知します</p>
    <form>
      <input type="button" value="通知を消す" id="hide">
    </form>
  </body>
</html>
```

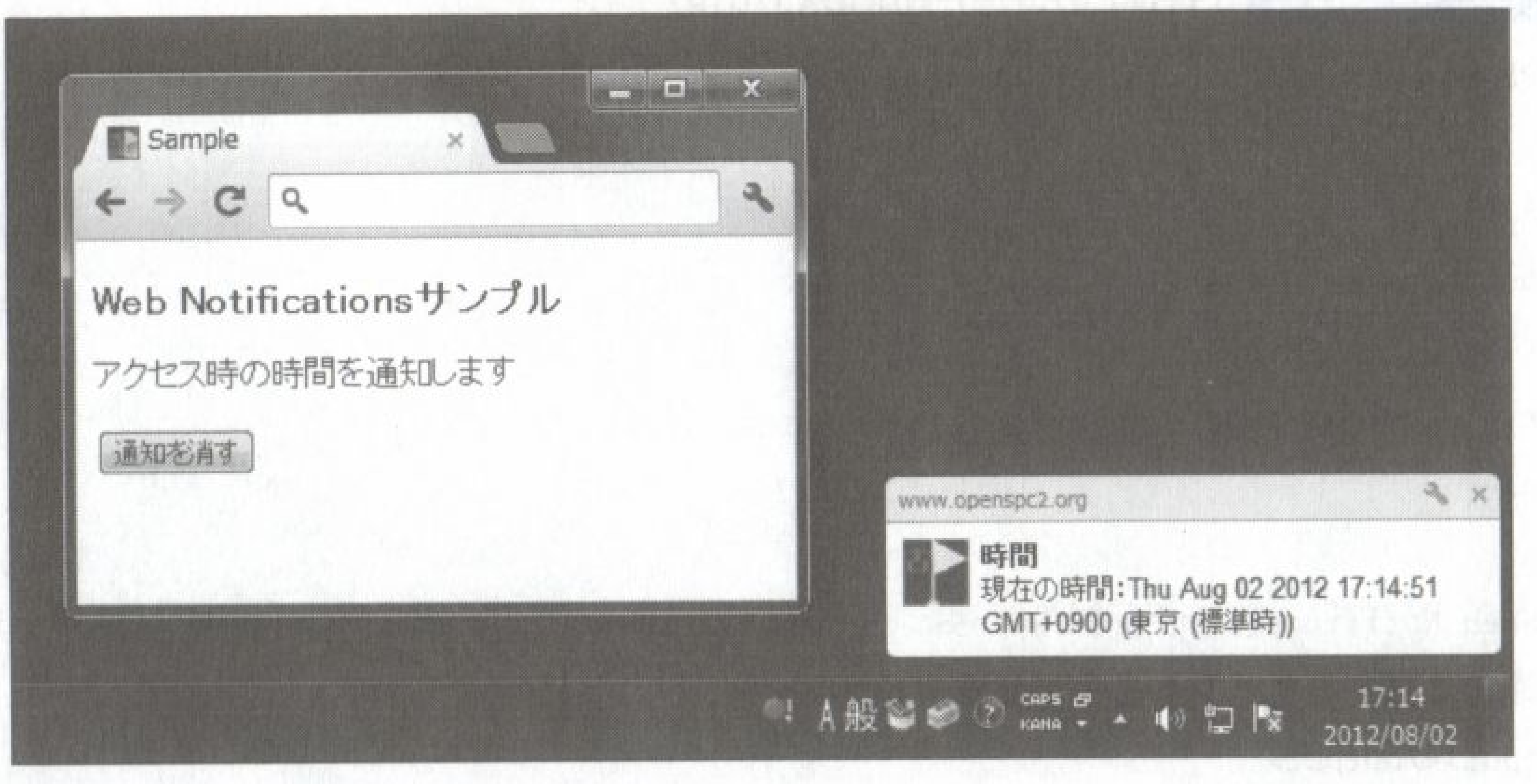
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // Notificationオブジェクトを入れる変数
  var popup = null;
  // Google Chrome, Safari 6で使用可能な場合に処理
  if (webkitNotifications){
    // 通知するメッセージを準備
    var message = "現在の時間 :"+(new Date());
    // 通知を行う
    popup = webkitNotifications.createNotification("icon.png", "時間", message);
    // 画面に表示
    popup.show();
  }
  // ボタンにイベントを割り当て
  document.getElementById("hide").addEventListener("click", function(){
    // Notificationオブジェクトが存在する場合のみ処理
    if (popup){
      // 通知を消す
      popup.cancel();
    }
  }, false);
}, false);
```

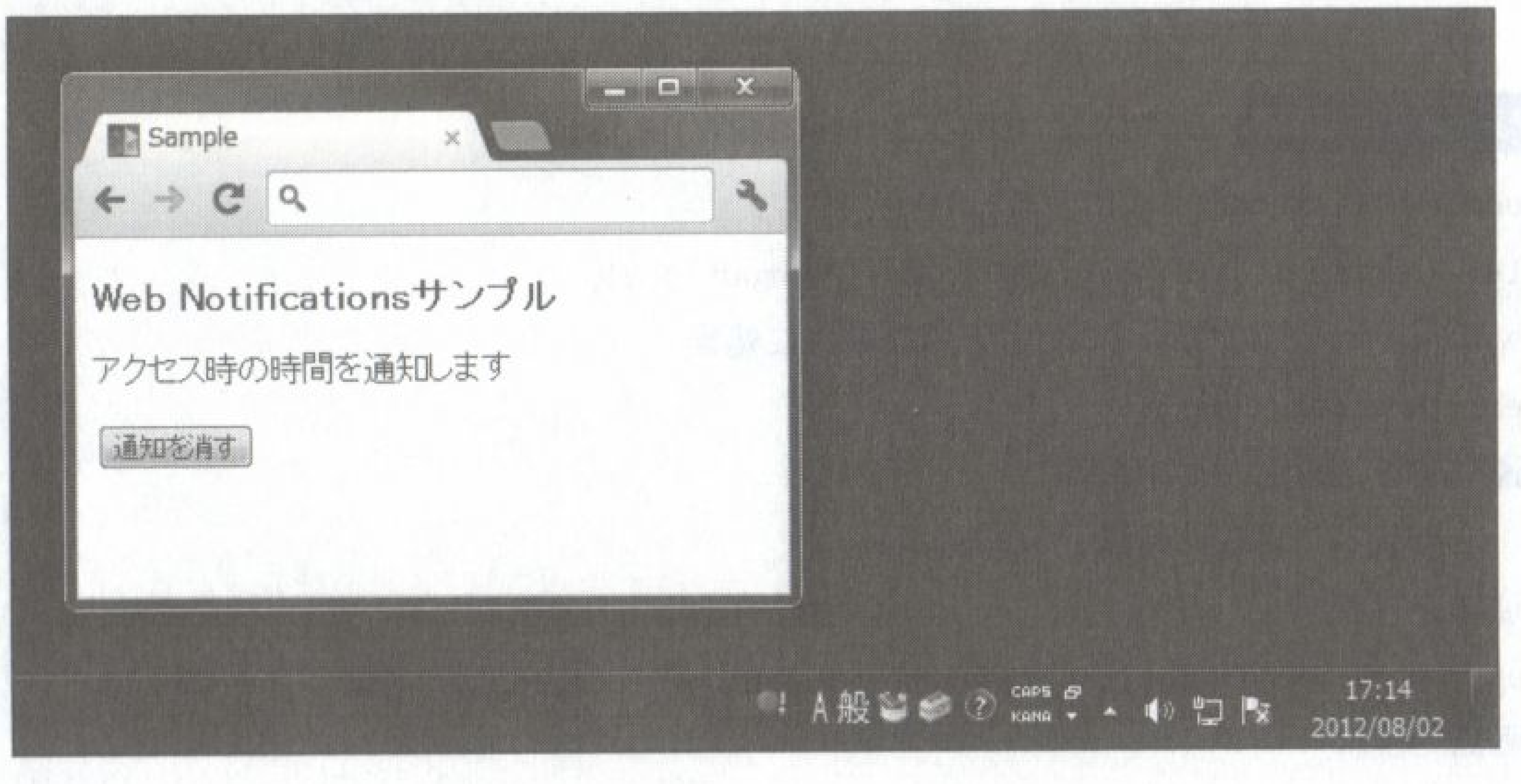
HINT

Safari 6の場合、OS X 10.8(Mountain Lion)上でのみ動作します。また、Google Chromeで動作を確認する場合は、設定を変更する必要があります。それぞれ、871ページのCOLUMNを参照してください。

ブラウザでindex.htmlを表示し、ページが読み込まれると、その時点でのアクセス時間が通知されます。



ボタンをクリックすると、通知が消えます。



ONEPOINT 通知を消すには「cancel()」メソッドを使う

画面に表示されている通知を消すには、「Notification」オブジェクトの「cancel()」メソッドを使います。「cancel()」メソッドが呼び出されると、通知はすぐに消去されます。なお、消去されるのは対応する通知だけで、すでに表示されている他の通知が消えることはありません。

なお、Safari 6では通知を消せない場合があります。

関連項目 ▶▶▶

- 通知可能かどうか調べる ………… p.869
- 通知する ………… p.872
- 通知に関するイベントを処理する ………… p.876

SECTION-370

通知に関するイベントを処理する

ここでは、Web Notifications APIを利用した通知に関するイベントを処理する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>Web Notificationsサンプル</h3>
    <p>通知に関するイベントを表示します</p>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

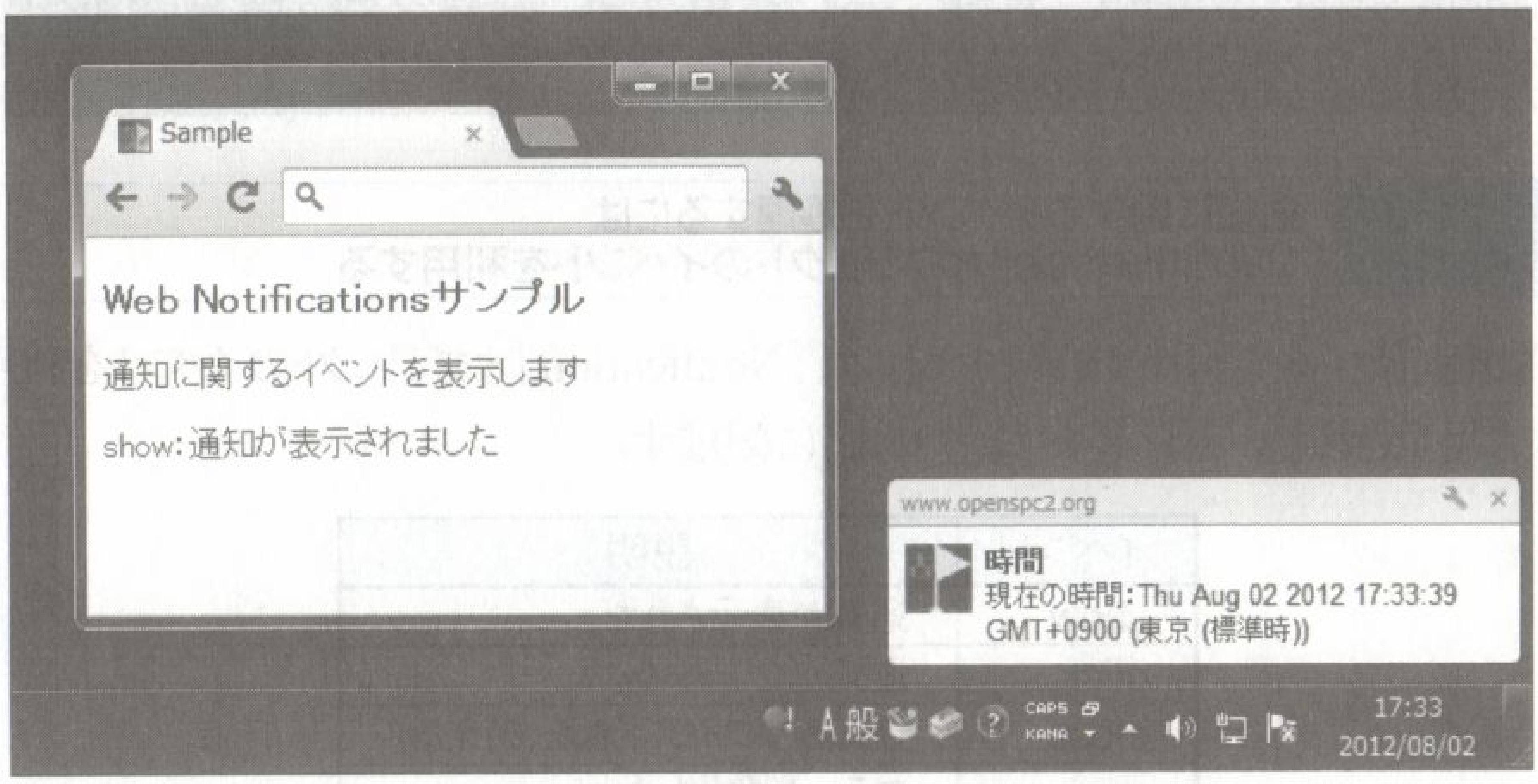
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // Google Chrome, Safari 6で使用可能な場合に処理
  if (webkitNotifications){
    // 通知するメッセージを準備
    var message = "現在の時間 :"+(new Date());
    // 通知を行う
    popup = webkitNotifications.createNotification("icon.png", "時間", message);
    // 画面に表示
    popup.show();
    // 通知が表示された場合のイベントを設定
    popup.onshow = function(){
      ele.innerHTML += "show:通知が表示されました<br>";
    }
    // 通知がクリックされた場合のイベントを設定
    popup.onclick = function(){
      ele.innerHTML += "click:通知がクリックされました<br>";
    }
    // 通知が消えた場合のイベントを設定
    popup.onclose = function(){
      ele.innerHTML += "close:通知がクローズされました<br>";
    }
}
```

```
}, false);
```

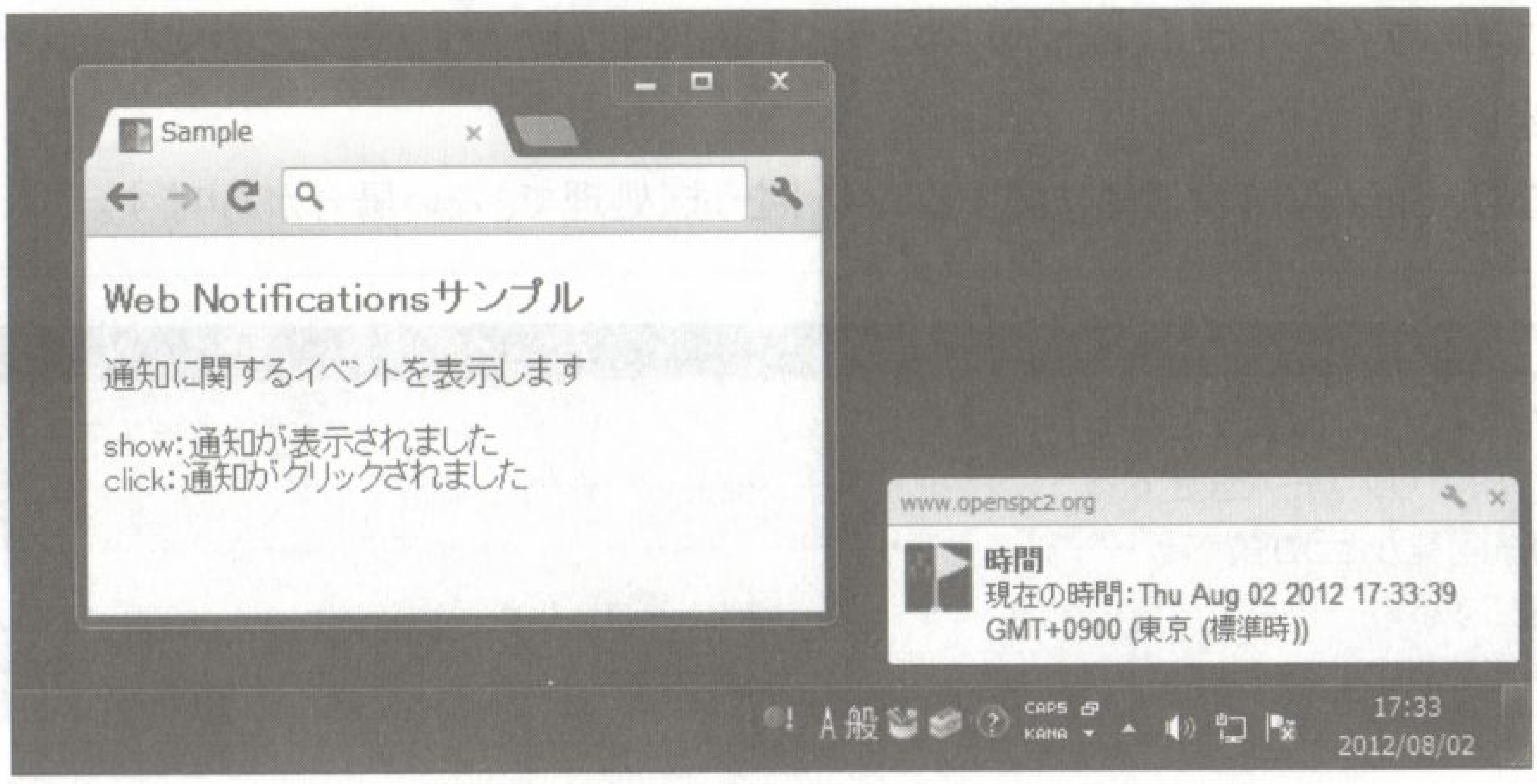
HINT

Safari 6の場合、OS X 10.8(Mountain Lion)上でのみ動作します。また、Google Chromeで動作を確認する場合は、設定を変更する必要があります。それぞれ、871ページのCOLUMNを参照してください。

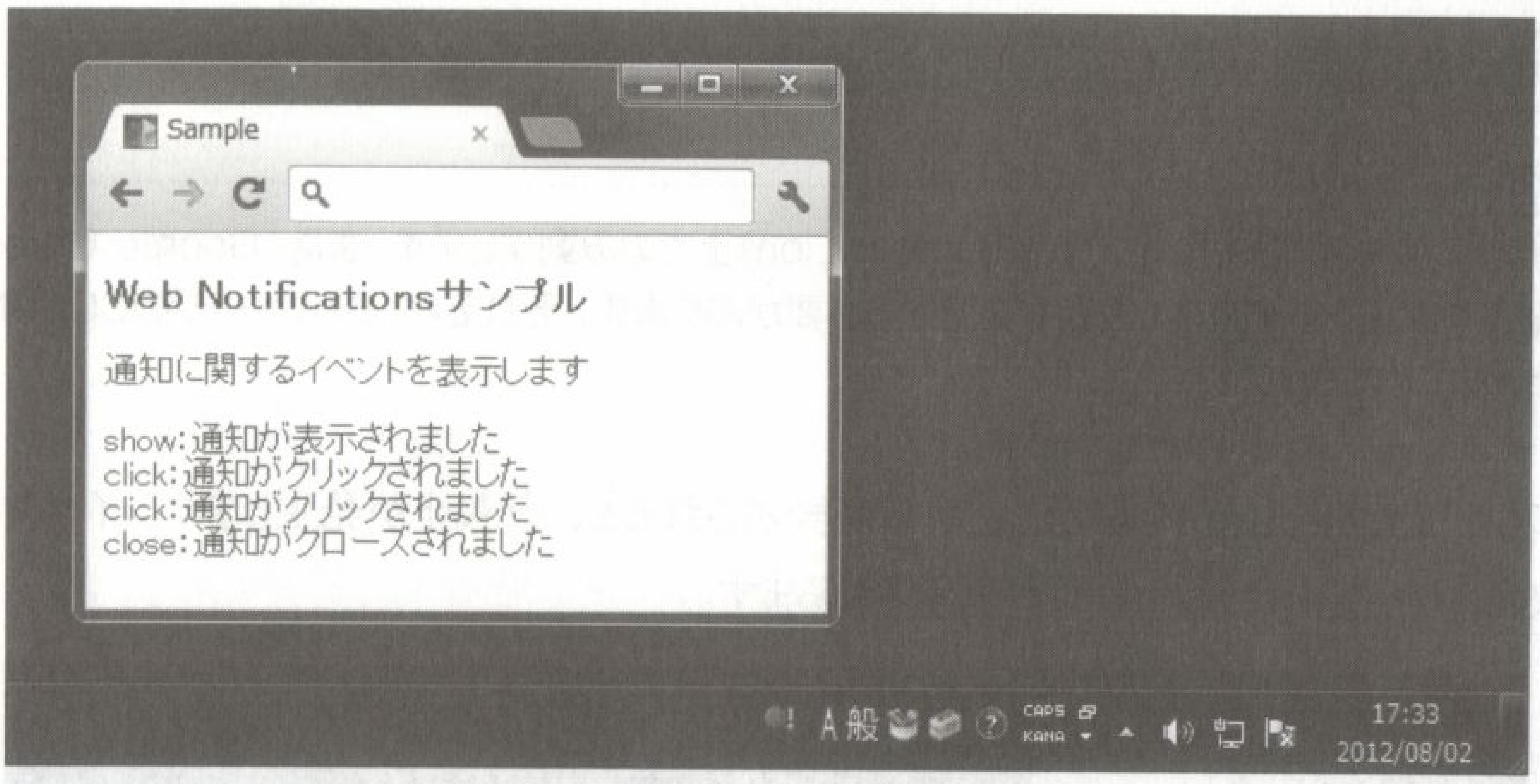
ブラウザでindex.htmlを表示し、通知が表示されると、イベントが発生します。イベントが発生すると、ページ上にメッセージが表示されるます。



通知上でクリックするとイベントが発生し、ページ上にメッセージが表示されます。



通知を閉じるとイベントが発生し、ページ上にメッセージが表示されます。



ONEPOINT 通知に関するイベントを処理するには「Notification」オブジェクトのイベントを利用する

通知に関するイベントを処理するには、「Notification」オブジェクトのイベントを利用します。用意されているイベントは、次のようになります。

イベント	説明
show	通知が表示された
click	通知がクリックされた
close	通知がクローズされた(消えた)
error	エラーが発生した
display	showと同じ(現在は仕様外)

これらのイベントは、「onshow」のように、「on」を付けたプロパティにイベントハンドラを設定します。

なお、Safari 6では通知に関するイベントをうまく処理できない場合があります。

関連項目 ▶▶▶

●イベントハンドラを設定する ……… p.396
●イベントリスナーを設定する ……… p.398
●通知可能かどうか調べる ……… p.869
●通知する ……… p.872
●通知を消す ……… p.874

CHAPTER 14
スタイルシート

SECTION-371

スタイルシートを生成する/読み込む

ここでは、すでに存在するページ内において新たにスタイルシートを生成する方法と、CSSファイルを読み込んでスタイルを追加する方法について解説します。

SAMPLE CODE HTML(index.html)

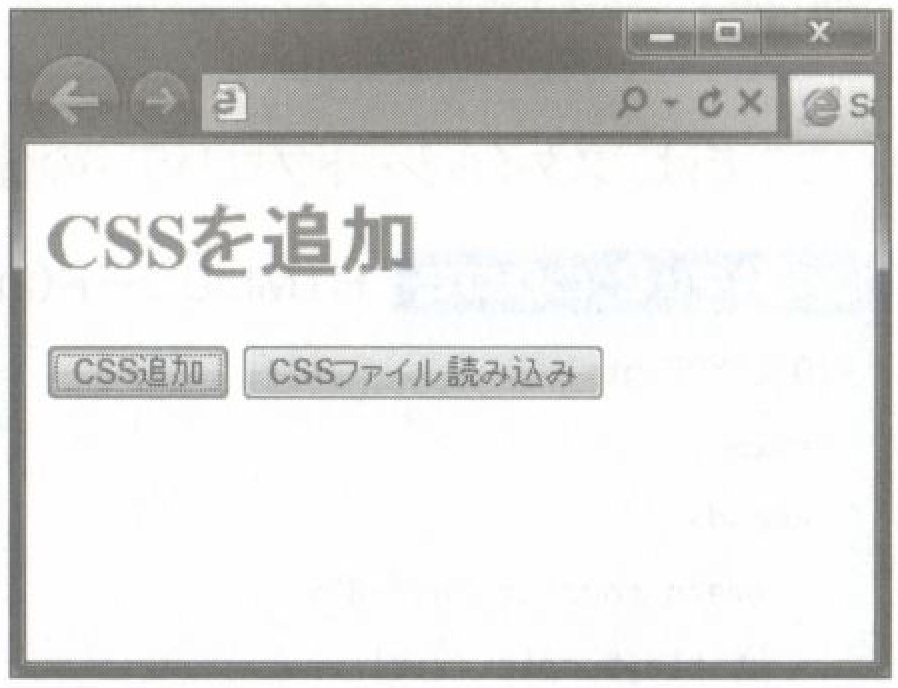
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>CSSを追加</h1>
    <form>
      <input type="button" value="CSS追加" id="addBtn">
      <input type="button" value="CSSファイル読み込み" id="loadBtn">
    </form>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

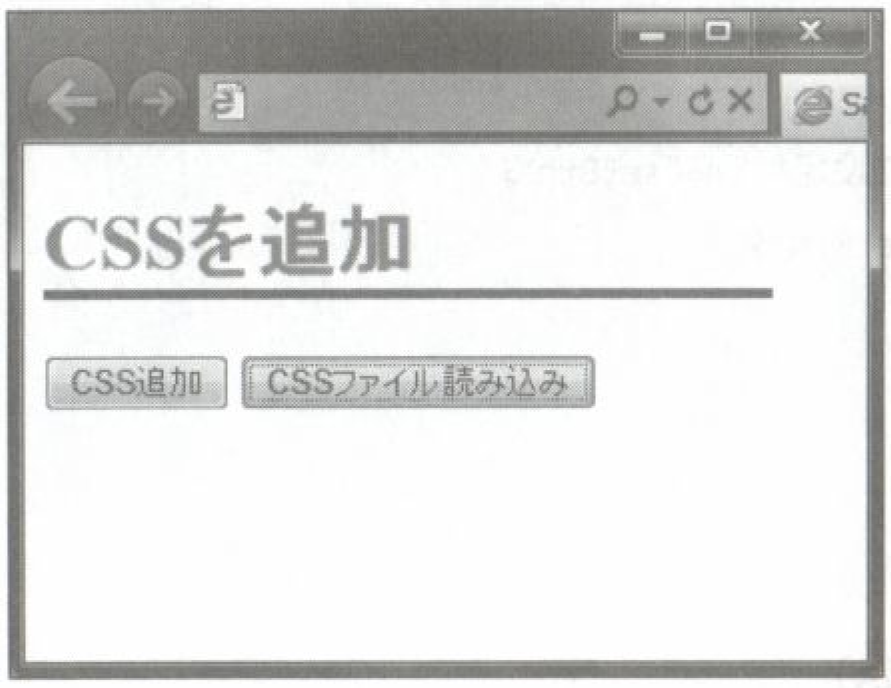
```
window.onload = function(){
  // CSS追加ボタンにイベントを設定
  document.getElementById("addBtn").addEventListener("click", function(){
    // style要素を生成
    var cssEle = document.createElement("style");
    // CSSのプロパティと値を設定
    cssEle.innerText = cssEle.textContent = "h1 { color: red; }";
    // head要素の末尾に追加(scoped属性なし)
    document.getElementsByTagName("head")[0].appendChild(cssEle);
  }, false);
  // CSSファイル読み込みボタンにイベントを設定
  document.getElementById("loadBtn").addEventListener("click", function(){
    // link要素を生成
    var linkEle = document.createElement("link");
    // rel属性とCSSファイルのURLを設定
    linkEle.rel = "stylesheet";
    linkEle.href = "css/main.css";
    // head要素の末尾に追加
    document.getElementsByTagName("head")[0].appendChild(linkEle);
  }, false);
}
```

ブラウザでindex.htmlを表示し、「CSS追加」ボタンをクリックすると、「style」要素が作成されて見出しの文字の色が赤色になります。





また、「CSSファイル読み込み」ボタンをクリックすると、「link」要素が作成されてスタイルシートファイルが読み込まれ、見出しに青い下線が追加されます。



ONEPOINT スタイルシートを生成したり読み込んだりするには「document.createElement()」メソッドを使う

スタイルを新たに追加・作成するには、「document.createElement()」メソッドを使い、「style」要素を作成します。スタイルシートに設定する内容を「style」要素の「innerText」プロパティや「textContent」プロパティに設定します。

すでに用意されているCSSファイルを読み込む場合は、「document.createElement()」メソッドを使い、「link」要素を作成します。「rel」属性には「stylesheet」の文字を、「href」属性にはCSSファイルのURLを指定します。

「style」要素、「link」要素とも、「appendChild()」メソッドを使ってページ内に追加します。

なお、IE6以降では、「document.createStyleSheet()」メソッドを使って、CSSファイルを追加することができます。また、CSSの内容を示す「cssText」プロパティもあります。

SECTION-372

スタイルシートプロパティの値を設定する

ここでは、スタイルシートプロパティの値を設定する方法について解説します。

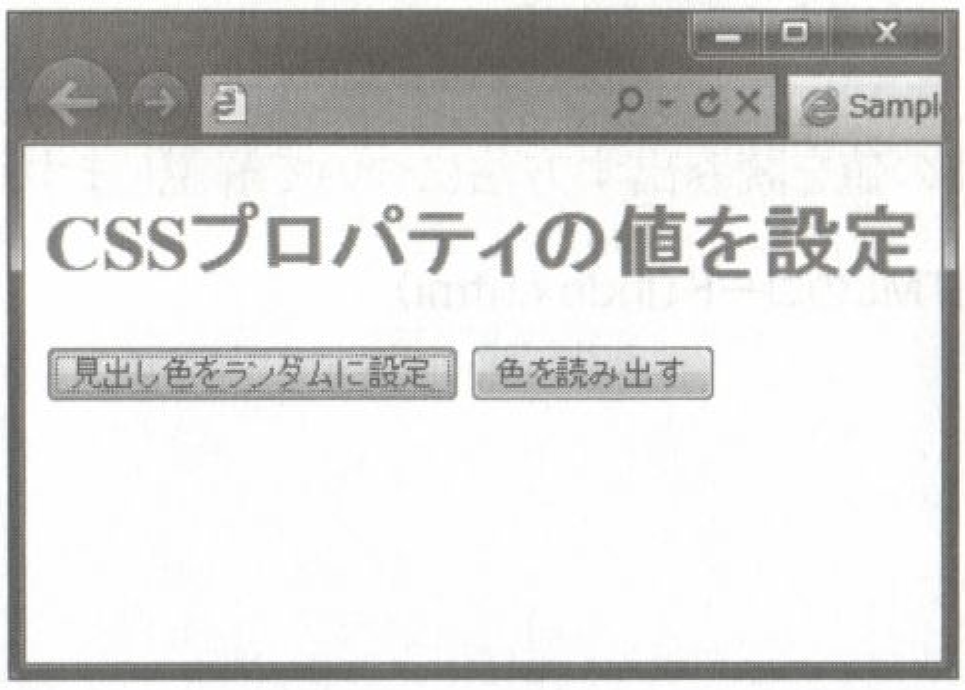
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>CSSプロパティの値を設定</h1>
    <form>
      <input type="button" value="見出し色をランダムに設定" id="setBtn">
      <input type="button" value="色を読み出す" id="getBtn">
    </form>
    <output></output>
  </body>
</html>
```

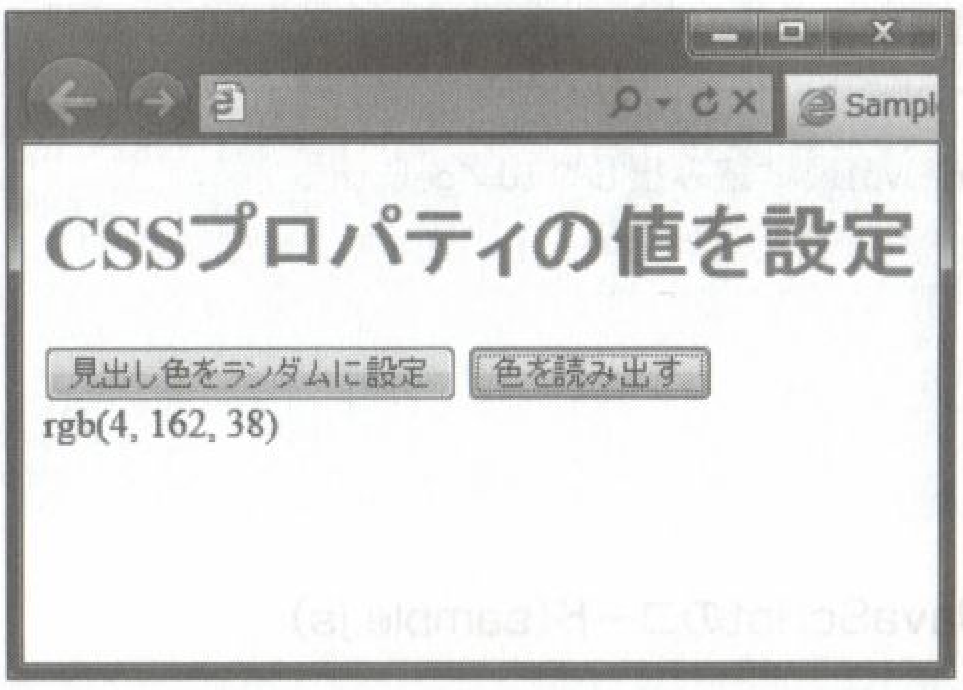
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("setBtn").addEventListener("click", function(){
    // h1(見出し)要素を読み出し
    var h1Ele = document.getElementsByTagName("h1")[0];
    // ランダムな色を算出
    var r = Math.floor(Math.random() * 255);
    var g = Math.floor(Math.random() * 255);
    var b = Math.floor(Math.random() * 255);
    var RGBcolor = "rgb("+r+","+g+","+b+")";
    // h1の文字色のプロパティを設定
    h1Ele.style.color = RGBcolor;
  }, false);
  // ボタンにイベントを設定
  document.getElementById("getBtn").addEventListener("click", function(){
    var ele = document.getElementsByTagName("output")[0];
    // h1(見出し)要素を読み出し
    var h1Ele = document.getElementsByTagName("h1")[0];
    // h1の文字色を読み出して表示
    ele.innerHTML = h1Ele.style.color;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「見出し色をランダムに設定」ボタンをクリックすると、見出しの文字の色がランダムに変わります。



また、「色を読み出す」ボタンをクリックすると、見出しの文字の色が表示されます（初回を除く）。



ONEPOINT

スタイルシートプロパティの値を設定するには
要素の「style」オブジェクトの各プロパティに値を書き込む

スタイルシートプロパティの値を設定するには、要素の「style」オブジェクトの各プロパティに値を書き込みます。たとえば、文字の色であれば、「color」プロパティにカラー名やカラーコードを設定します。CSSプロパティで設定できる形式と同じです。また、背景色を表す「background-color」のように、CSSでは「-」（ハイフン）が入っている場合は「-」を削除し、直後の文字を大文字したものがJavaScriptでのプロパティ名になります。「background-color」の場合は、「backgroundColor」プロパティにアクセスすることで背景色を変更することができます。

「style」オブジェクトには、書き込んだ値が保持されています。書き込んでいない場合には、「style」オブジェクト内のプロパティを読み出しても期待するような結果を得ることはできません。

関連項目 ▶▶▶

● スタイルシートの値を読み出す p.884

スタイルシートの値を読み出す

ここでは、スタイルシートの値を読み出す方法について解説します。

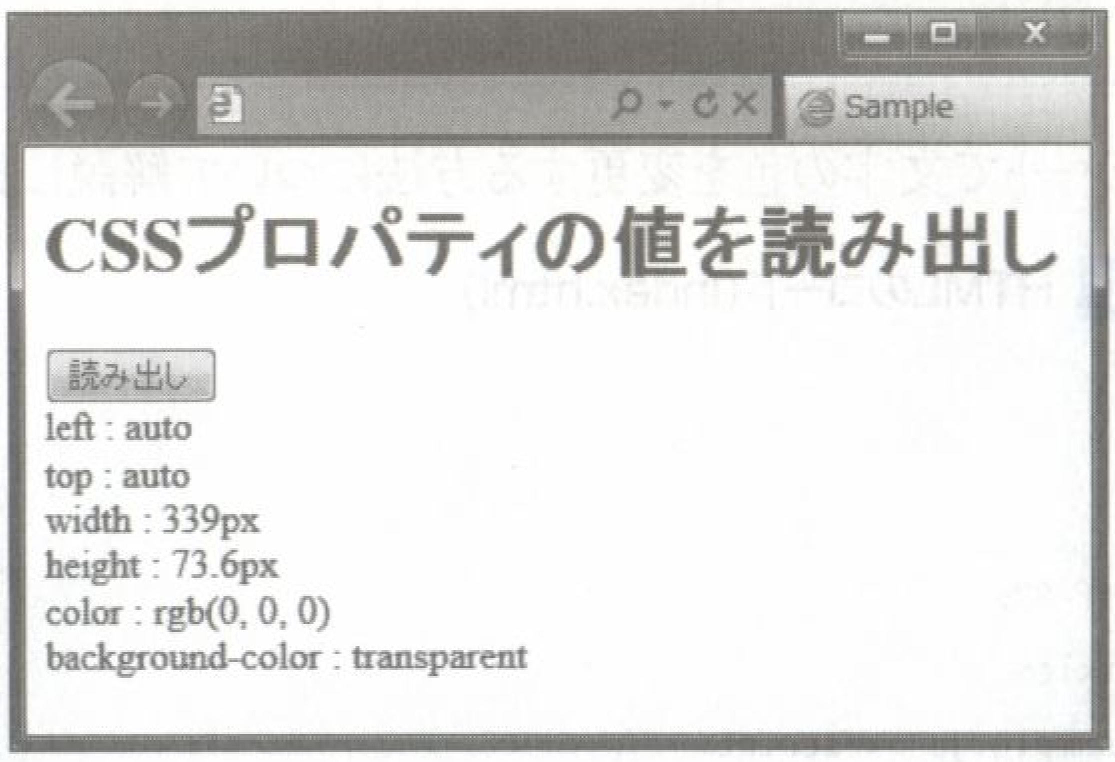
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>CSSプロパティの値を読み出し</h1>
    <form>
      <input type="button" value="読み出し" id="getBtn">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("getBtn").addEventListener("click", function(){
    var ele = document.getElementsByTagName("output")[0];
    // h1(見出し)要素を読み出し
    var h1Ele = document.getElementsByTagName("h1")[0];
    // h1の文字色を読み出して表示
    var data = document.defaultView.getComputedStyle(h1Ele, "");
    // 各プロパティを読み出し
    ele.innerHTML = "left : "+data.getPropertyValue("left");
    ele.innerHTML += "<br>top : "+data.getPropertyValue("top");
    ele.innerHTML += "<br>width : "+data.getPropertyValue("width");
    ele.innerHTML += "<br>height : "+data.getPropertyValue("height");
    ele.innerHTML += "<br>color : "+data.getPropertyValue("color");
    ele.innerHTML += "<br>background-color : "+data.getPropertyValue("background-color");
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「読み出し」ボタンをクリックすると、見出しのプロパティが読み出されて表示されます。



ONEPOINT スタイルシートの値を読み出すには要素の「style」オブジェクトの各プロパティを参照する

スタイルシートの値を読み出すには、要素の「style」オブジェクトの各プロパティを参照します。ただし、「style」オブジェクトの各プロパティを参照した場合、実際に設定されている値とは異なる場合があります。

実際に適用されている値を取得したい場合は、「document.defaultView.getComputedStyle()」メソッドを使います。最初のパラメータには要素を、2番目のパラメータには疑似要素を指定します。疑似要素を指定する必要がない場合は、空文字(または「null」)を指定します。

また、「document.defaultView.getComputedStyle()」メソッドではなく、「window.getComputedStyle()」と記述することもできます。

関連項目 ▶▶▶

●スタイルシートプロパティの値を設定する …… p.882

文字の色を変更する

ここでは、スタイルシートで文字の色を変更する方法について解説します。

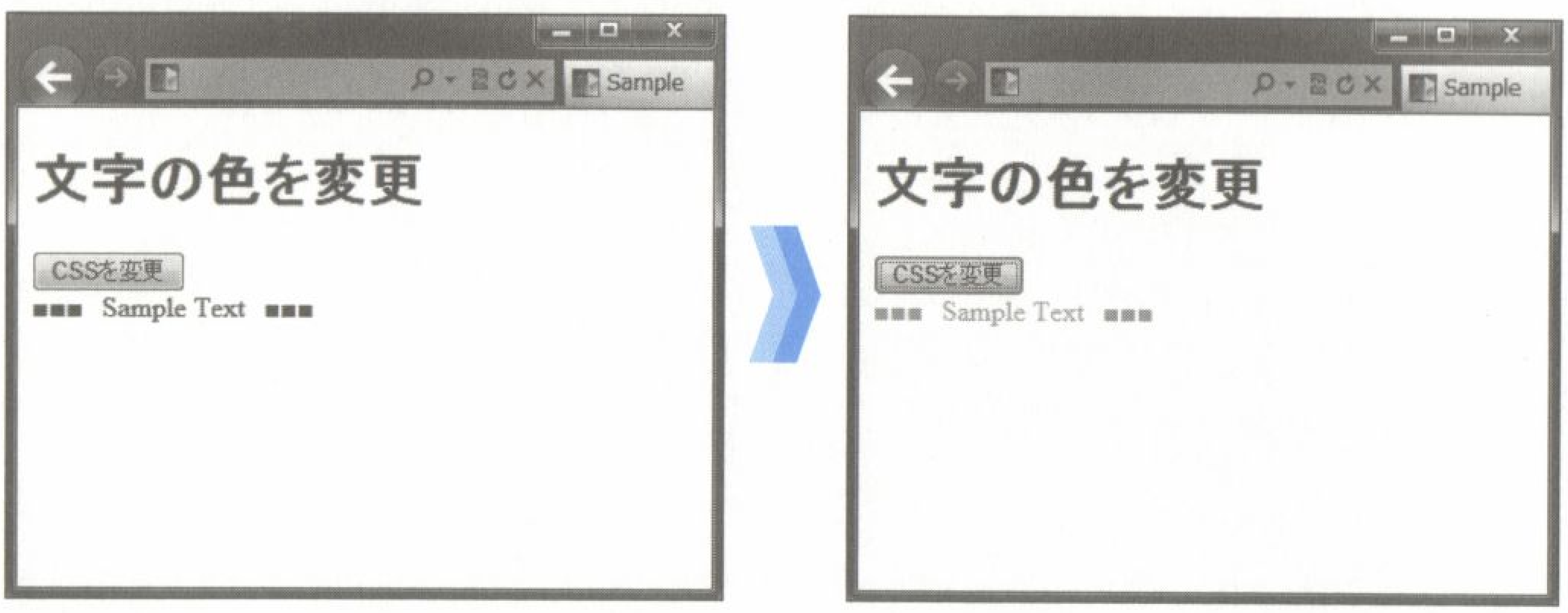
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の色を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■■■  Sample Text  ■■■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 文字色を設定
    divElement.style.color = "rgba(255, 0, 0, 0.5)";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の色が変更されます。



ONEPOINT 文字の色を変更するにはスタイルシートの「color」プロパティを使う

文字の色を変更するには、スタイルシートの「color」プロパティに色を示すカラー名やカラーコードを指定します。カラー名は、「red」や「blue」などの文字列を指定することができます。また、RGBでの指定やHSL指定も可能です。指定可能なカラー形式は、次の表のようになります。

形式	説明
カラー名	「red」や「green」など、あらかじめ定義されている文字列
#rrggbb	16進数6桁でRGBの輝度を指定(00〜ff)
#rgb	16進数3桁でRGBの輝度を指定(0〜f)
rgb(赤, 緑, 青)	0〜255までの輝度を指定。%での指定も可能
rgba(赤, 緑, 青, 不透明度)	0〜255までの輝度と、0.0〜1.0までの不透明度を指定
hsl(色相, 彩度, 輝度)	色相(0〜360)、彩度(%)、輝度(%)を指定
hsla(色相, 彩度, 輝度, 不透明度)	色相(0〜360)、彩度(%)、輝度(%)と、0.0〜1.0までの不透明度を指定
transparent	透明。「transparent」の文字をそのまま指定

詳しくは、次のURLを確認してください。

- CSS3 Color

  URL http://www.w3.org/TR/css3-color/

IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-375

文字のサイズを変更する

ここでは、スタイルシートで文字のサイズを変更する方法について解説します。

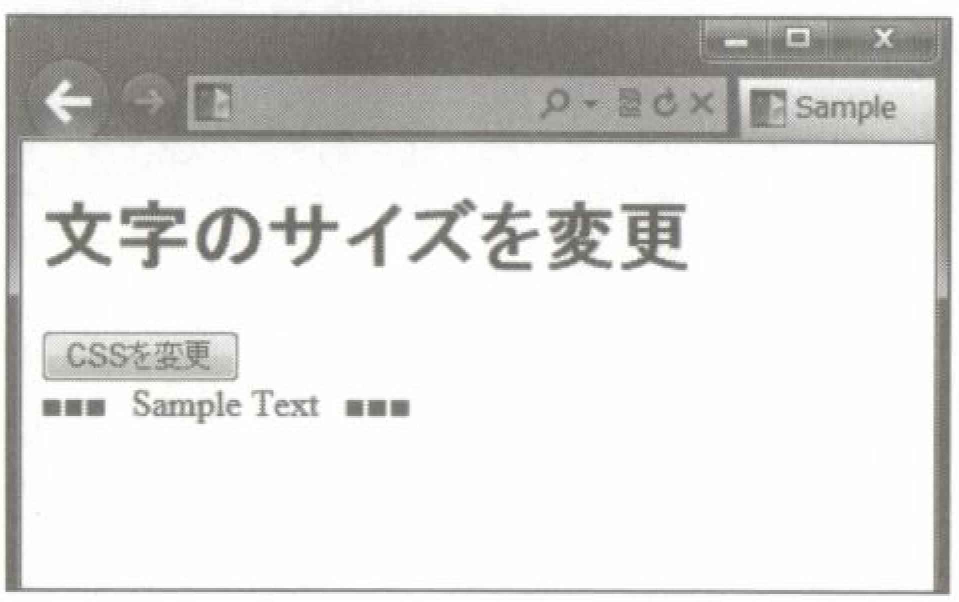
SAMPLE CODE HTMLのコード(index.html)

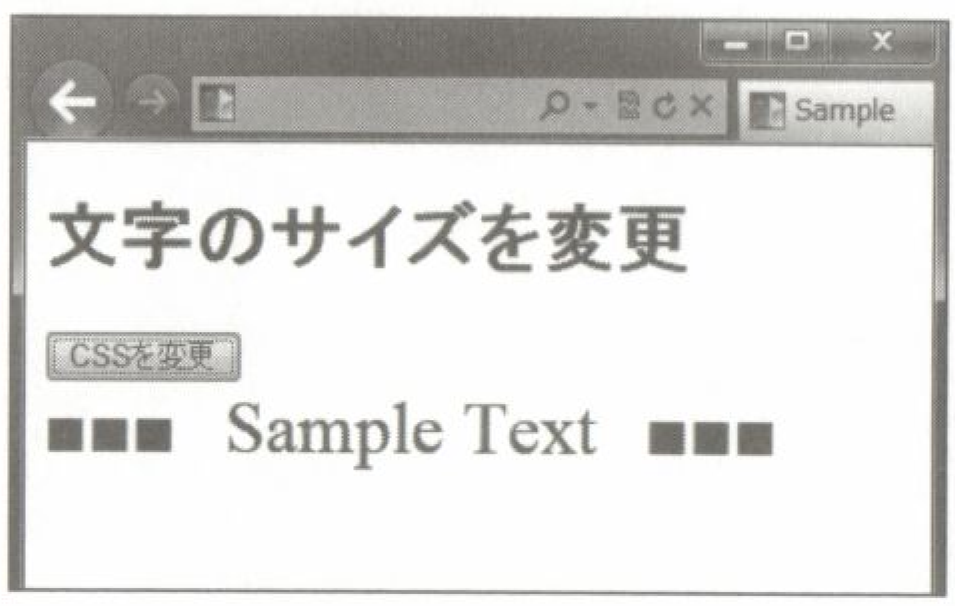
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字のサイズを変更</h1>
    <form><input type="button" value="CSSを変更" id="change"></form>
    <div>■■■  Sample Text  ■■■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    var divElement = document.getElementsByTagName("div")[0];  // div要素を読み出し
    divElement.style.fontSize = "24pt";  // 文字サイズを設定
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字のサイズが変更されます。





ONEPOINT 文字のサイズを変更するには「fontSize」プロパティを使う

文字のサイズを変更するには、スタイルシートの「fontSize」プロパティにサイズを示す値を設定します。値は単位付きの文字列になります。つまり、「"24pt"」とすると文字サイズは24ptに、「"120%"」とすると文字サイズは120%サイズになります。

文字の書体を変更する

ここでは、スタイルシートで文字の書体(フォント)を変更する方法について解説します。

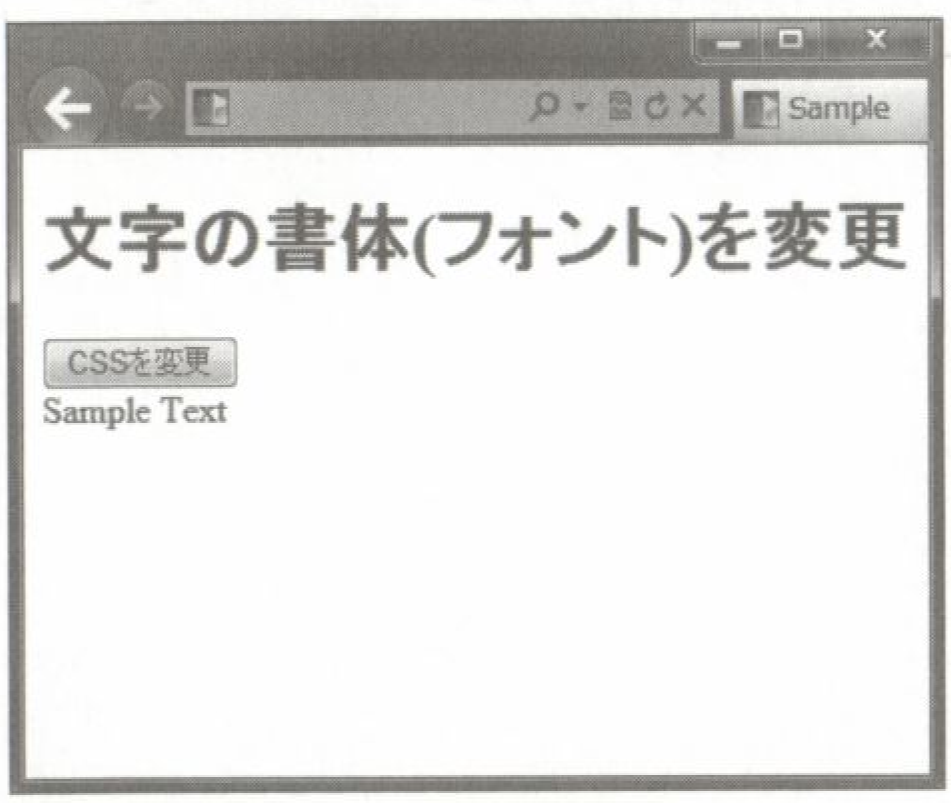
SAMPLE CODE HTMLのコード(index.html)

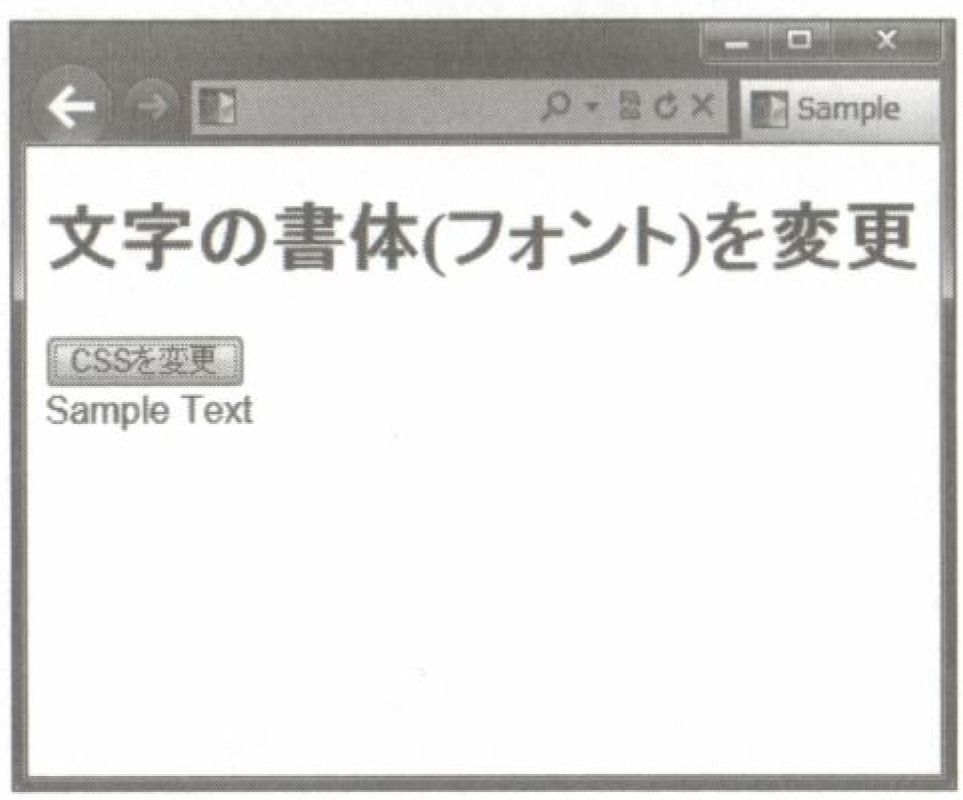
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の書体(フォント)を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 書体(フォント)を設定
    divElement.style.fontFamily = "Helvetica";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の書体が変更されます。





ONEPOINT 文字の書体を変更するには
スタイルシートの「fontFamily」プロパティを使う

文字の書体を変更するには、スタイルシートの「fontFamily」プロパティにフォント名を指定します。フォント名は、「Times」や「Helvetica」の文字列を指定します。書体によっては文字の太さを示す「bold」が付いている場合があります。たとえば、「Times-bold」のようなフォント名になっています。文字の太さや形状を示すフォント名を指定した場合、Mac版のSafariとGoogle Chromeでは反映されますが、他のOSやブラウザでは反映されません。文字の太さを指定する場合には、「fontWeight」プロパティに設定します。

COLUMN iOS 5でのフォント

iOS 5では、次のフォントがデフォルトで入っています。

- Academy Engraved LET
- American Typewriter
- Apple Color Emoji
- AppleGothic
- Arial
- Arial Hebrew
- Arial Rounded MT Bold
- Bangla Sangam MN
- Baskerville
- Bodoni 72
- Bodoni 72 Oldstyle
- Bodoni 72 Smallcaps
- Bodoni Ornaments
- Bradley Hand
- Chalkboard SE
- Chalkduster
- Cochin
- Copperplate
- Courier
- Courier New
- Devanagari Sangam MN
- Didot
- Euphemia UCAS
- Futura
- Geeza Pro
- Georgia
- Gill Sans
- Gujarati Sangam MN
- Gurmukhi MN
- Heiti SC
- Heiti TC
- Helvetica
- Helvetica Neue
- Hiragino Kaku Gothic ProN
- Hiragino Mincho ProN
- Hoefler Text
- Kailasa
- Kannada Sangam MN
- Malayalam Sangam MN
- Marion
- Marker Felt
- Noteworthy
- Optima
- Oriya Sangam MN
- Palatino
- Papyrus
- Party LET
- Sinhala Sangam MN
- Snell Roundhand
- Tamil Sangam MN
- Telugu Sangam MN
- Thonburi
- Times New Roman
- Trebuchet MS
- Verdana
- Zapf Dingbats
- Zapfino

詳細については、次のURLを参照してください。

URL http://support.apple.com/kb/HT4980?viewlocale=ja_JP&locale=ja_JP

文字の太さを変更する

ここでは、スタイルシートで文字の太さを変更する方法について解説します。

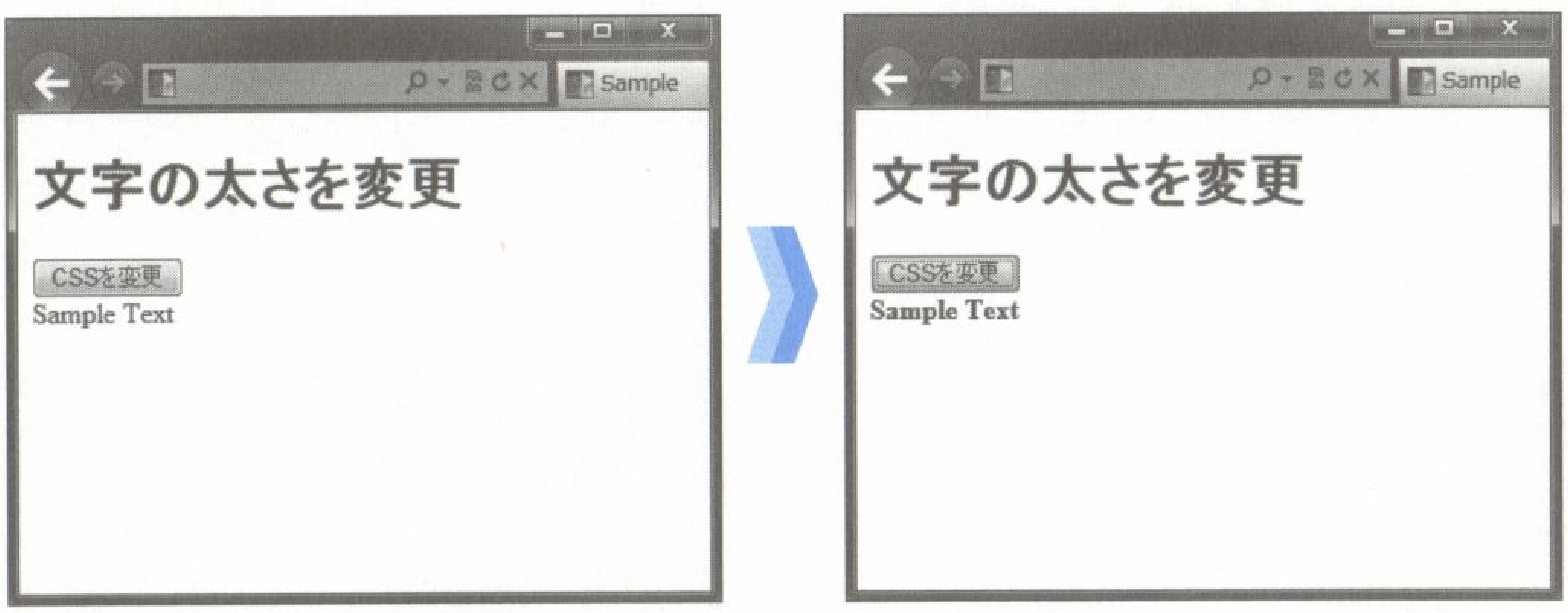
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の太さを変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 太さを設定
    divElement.style.fontWeight = 900;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の太さが変更されます。



ONEPOINT 文字の太さを変更するには スタイルシートの「fontWeight」プロパティを使う

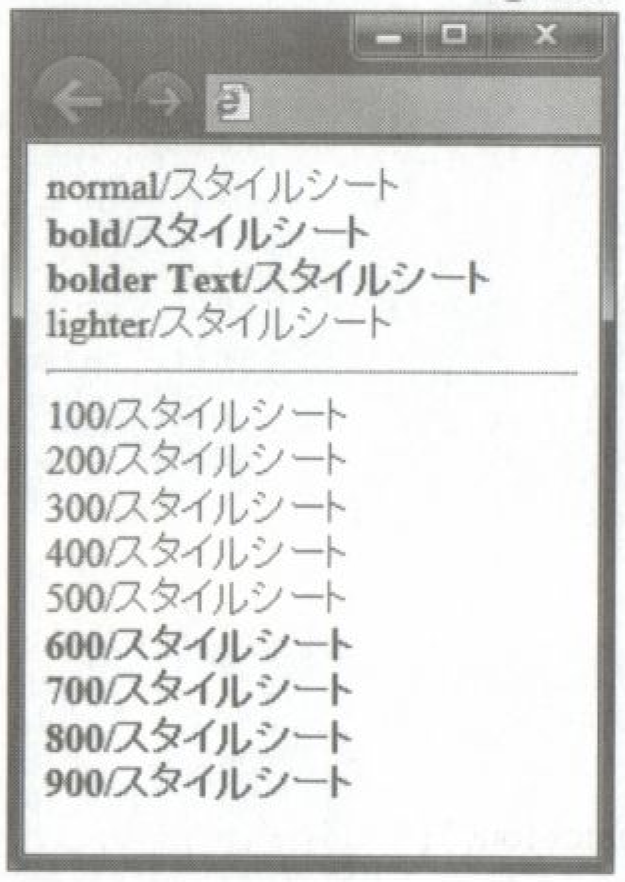
文字の太さを変更するには、スタイルシートの「fontWeight」プロパティに値を指定します。値は「100」～「900」までの数値になります。数値は値が大きくなるほど文字は太くなります。ただし、書体によっては反映されない場合があります。

また、次の表に示す文字列を使って太さを指定することもできます。

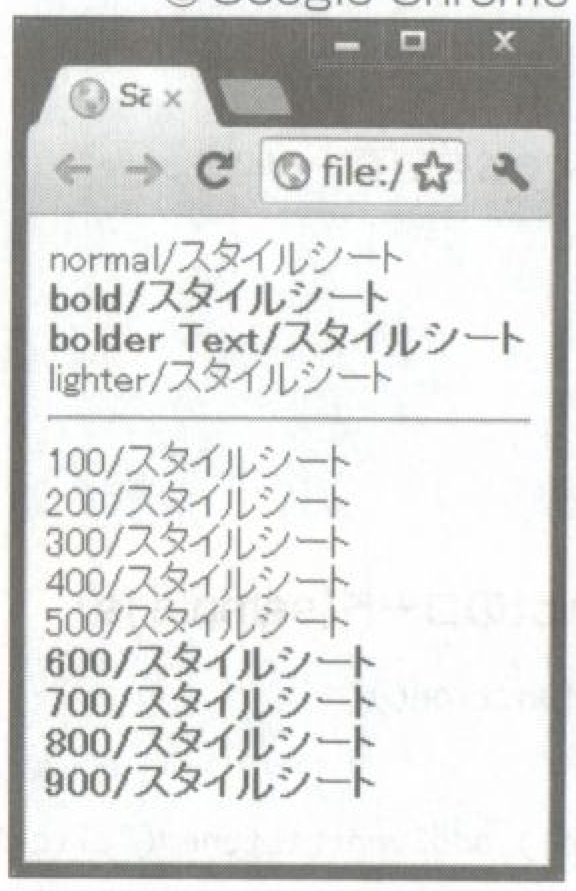
文字列	説明
normal	標準
bold	太く(数値で700を指定した場合と同じ)
bolder	より太く
lighter	より細く

それぞれの文字列と「100」～「900」を指定した結果は、次のようになります。

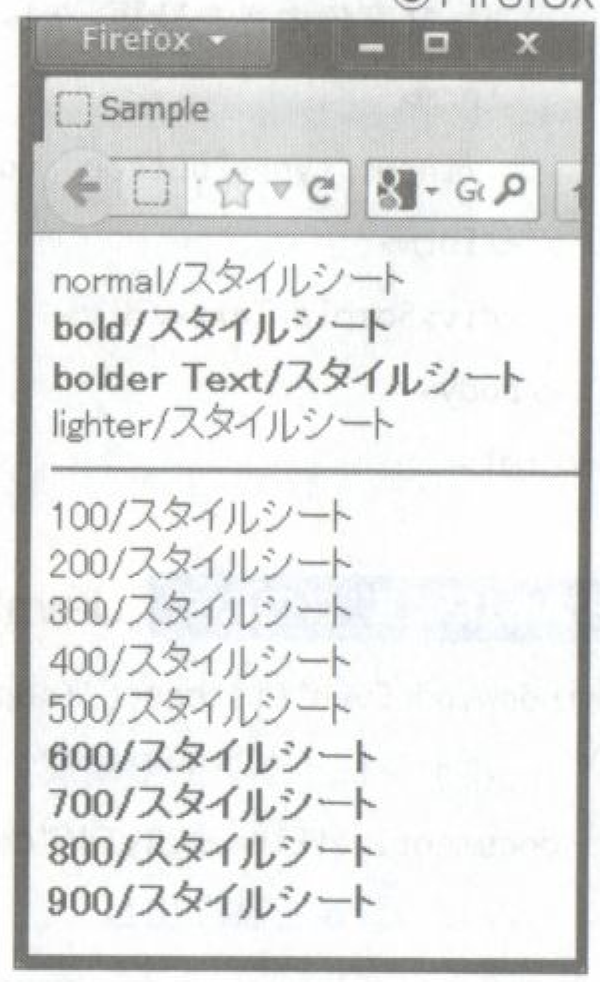
◉IE9



◉Google Chrome



◉Firefox



SECTION-378

文字のスタイルを変更する

ここでは、スタイルシートで文字のスタイル(イタリック体/ノーマル)を変更する方法について解説します。

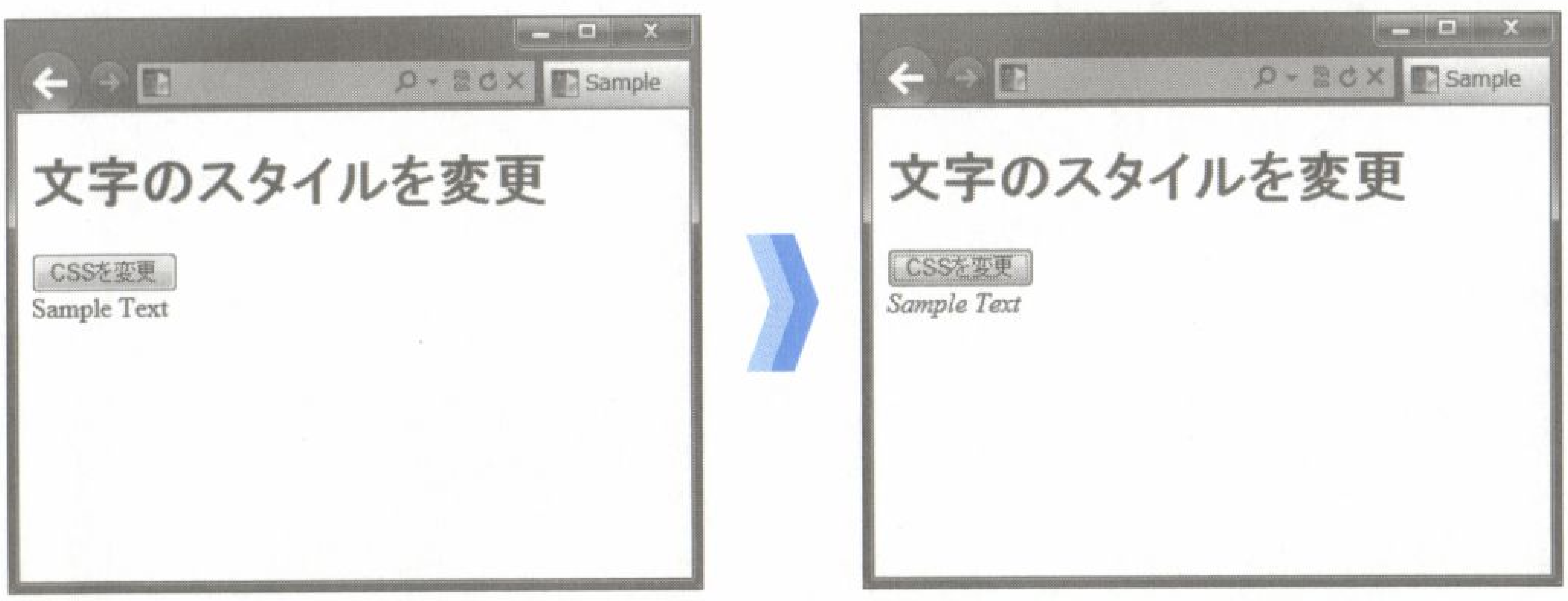
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字のスタイルを変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // スタイルを設定
    divElement.style.fontStyle = "italic";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字のスタイルが変更されます。



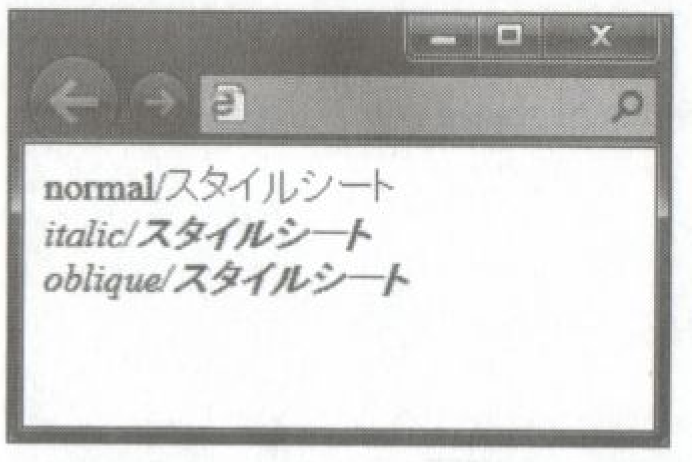
ONEPOINT

文字のスタイルを変更するには スタイルシートの「fontStyle」プロパティを使う

文字のスタイルを変更するには、スタイルシートの「fontStyle」プロパティに、次の表に示す文字列を設定します。なお、対応する斜めのフォントが存在しない場合、「italic」を指定しても「oblique」を指定しても同じ表示結果になります。

文字列	説明
normal	標準（正体）
italic	イタリック体
oblique	斜体

それぞれの文字列を指定した結果は、次のようになります。



文字の形状を変更する

ここでは、スタイルシートで文字の形状(スモールキャップス/ノーマル)を変更する方法について解説します。

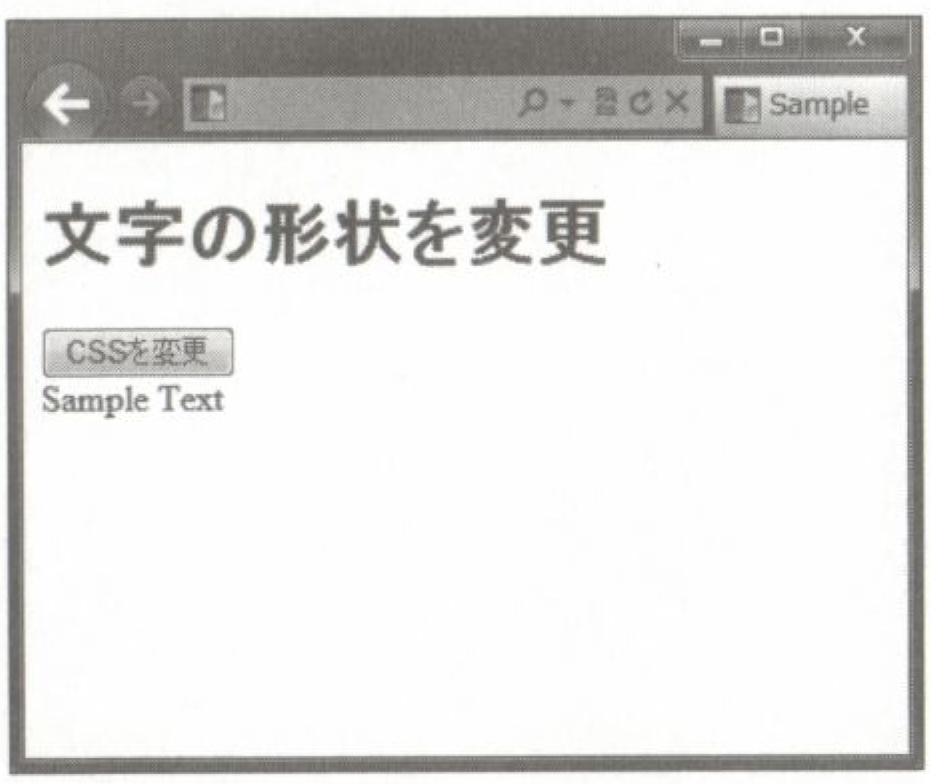
SAMPLE CODE HTMLのコード(index.html)

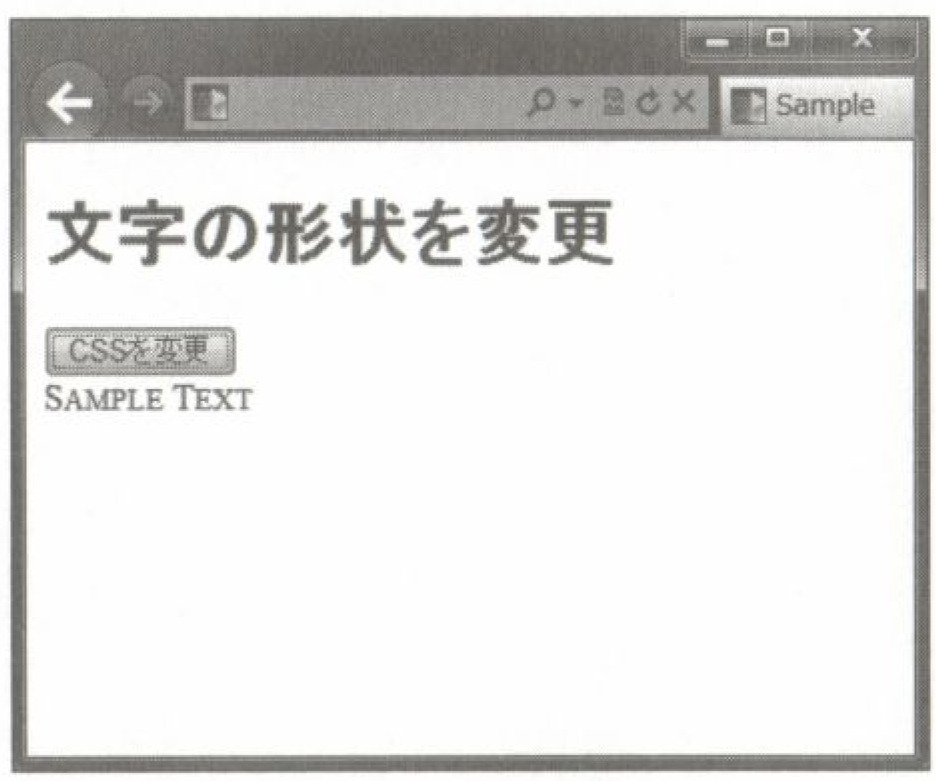
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の形状を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 形状を設定
    divElement.style.fontVariant = "small-caps";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の形状が変更されます。



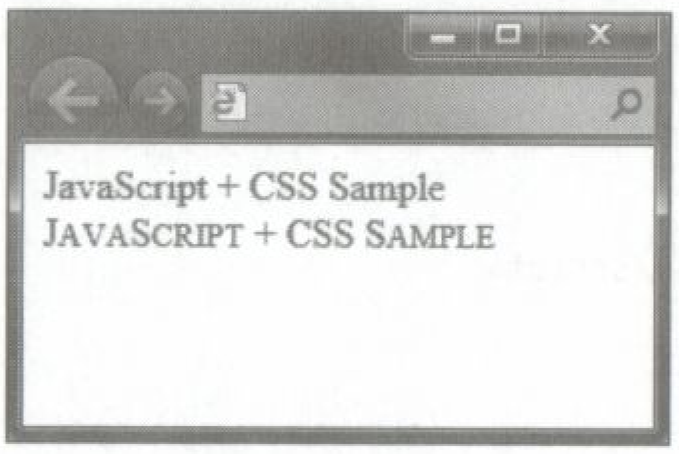


ONEPOINT 文字の形状を変更するには スタイルシートの「fontVariant」プロパティを使う

文字の形状（スモールキャップス/ノーマル）を変更するには、スタイルシートの「fontVariant」プロパティに、次の表に示す文字列を設定します。

文字列	説明
normal	標準
small-caps	小文字を英大文字にし、小さいサイズで表示する

それぞれの文字列を指定した結果は、次のようになります。



関連項目 ▶▶▶

● 英文字の大文字/小文字に変換する ……………… p.208
● 英文字の表示形式を指定する ……………… p.907

SECTION-380

文字の間隔を変更する

ここでは、スタイルシートで文字の間隔を変更する方法について解説します。

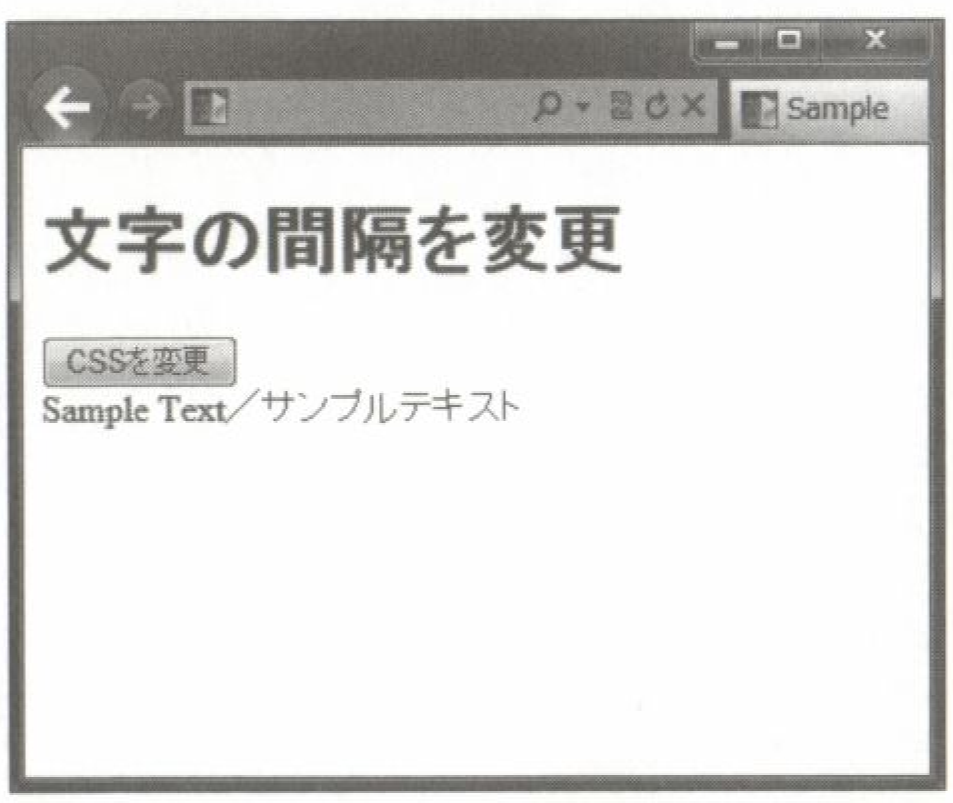
SAMPLE CODE HTMLのコード(index.html)

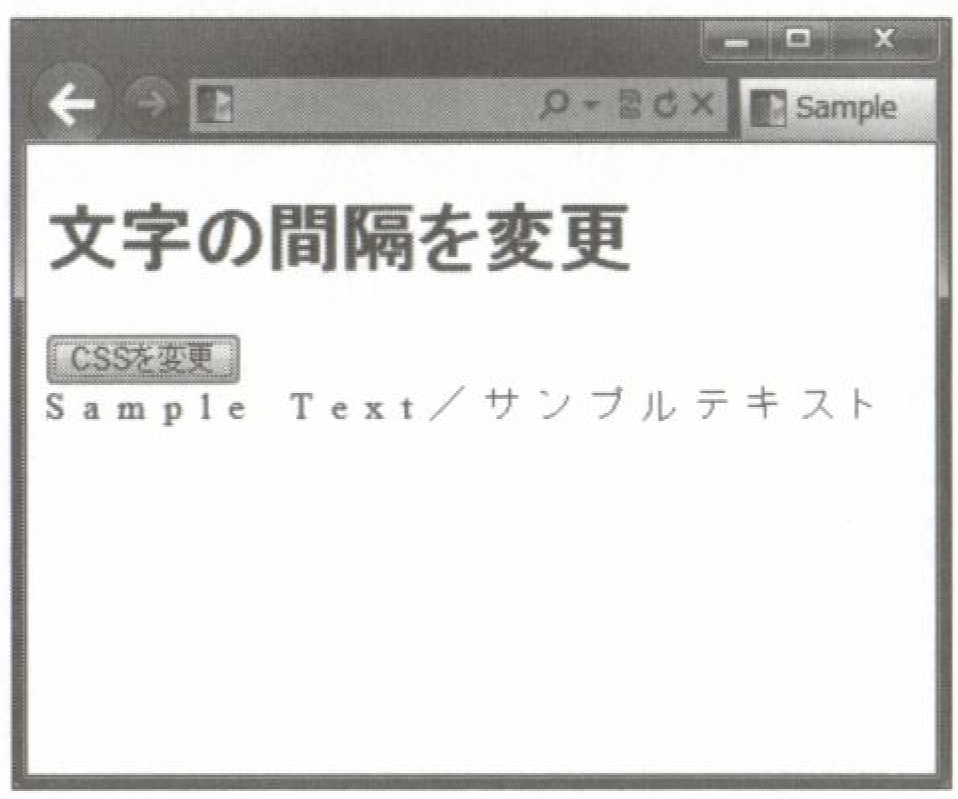
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の間隔を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text/サンプルテキスト</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 間隔を設定
    divElement.style.letterSpacing = "6pt";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の間隔が変更されます。





ONEPOINT

文字の間隔を変更するには
スタイルシートの「letterSpacing」プロパティを使う

文字と文字の間隔を変更するには、スタイルシートの「letterSpacing」プロパティに値を設定します。設定する値は、単位付きの文字列となります。たとえば、「"12pt"」とすると、文字と文字の間隔は12ptになります。

関連項目 ▶▶▶

● 単語の間隔を変更する …………………………………………………………………… p.899

SECTION-381

単語の間隔を変更する

ここでは、スタイルシートで単語の間隔を変更する方法について解説します。

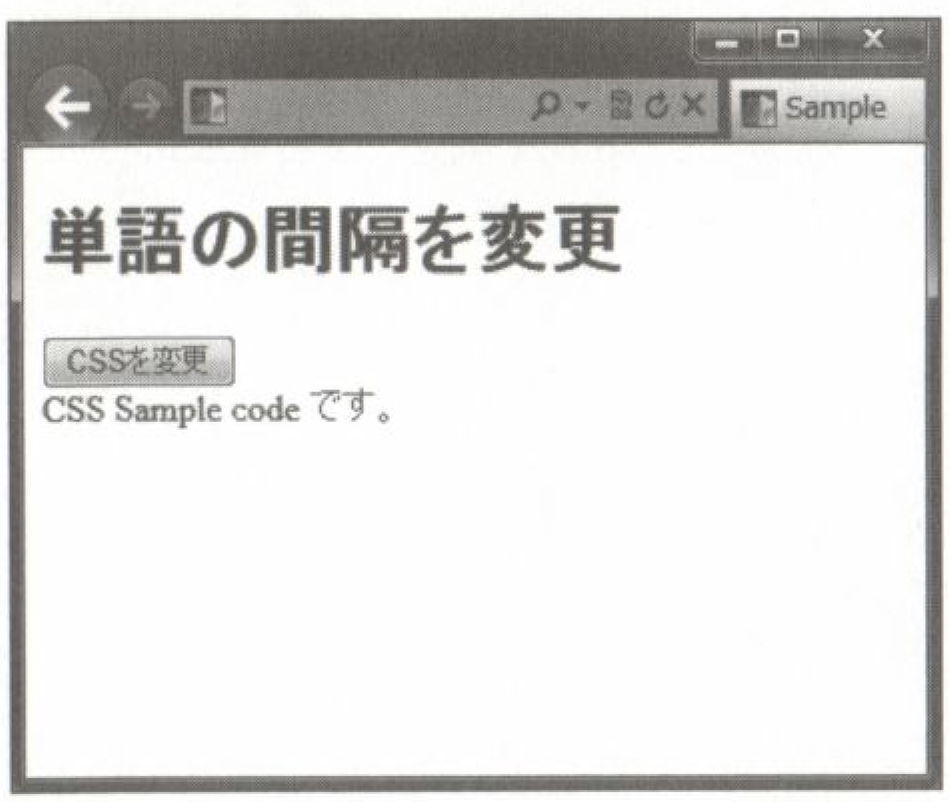
SAMPLE CODE HTMLのコード(index.html)

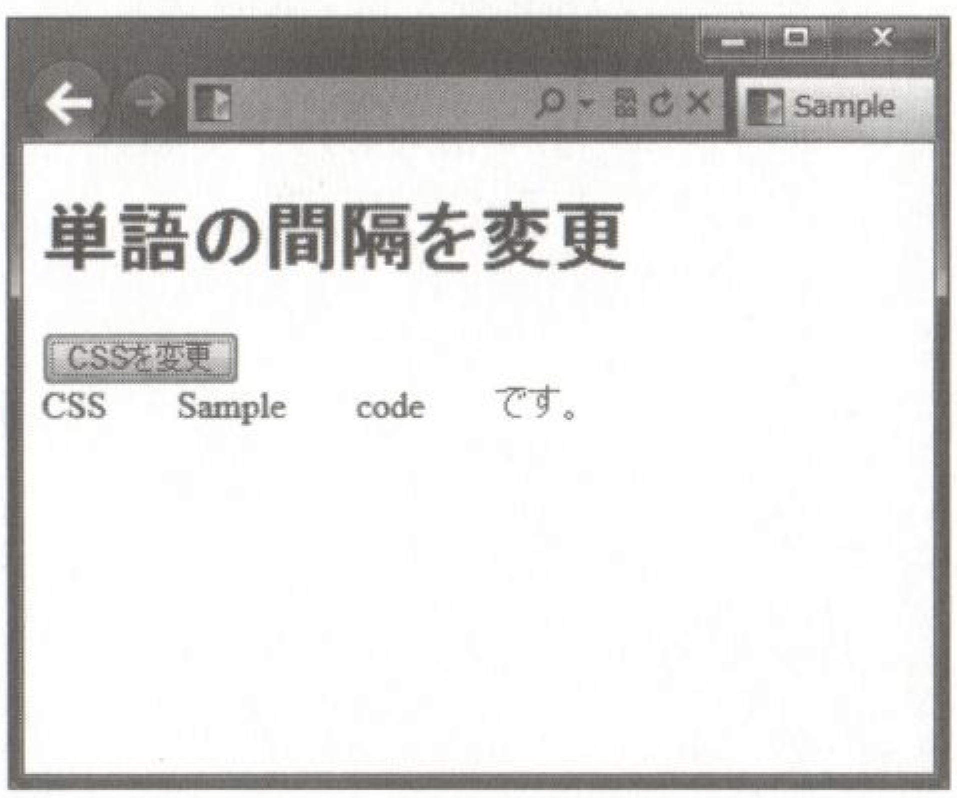
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>単語の間隔を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>CSS Sample code です。</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 間隔を設定
    divElement.style.wordSpacing = "20pt";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字の間隔が変更されます。





ONEPOINT 単語の間隔を変更するには スタイルシートの「wordSpacing」プロパティを使う

単語と単語の間隔を変更するにはスタイルシートの「wordSpacing」プロパティに値を設定します。設定する値は、単位付きの文字列になります。たとえば、「"12pt"」とすると文字と単語と単語の間隔は12ptになります。

関連項目 ▶▶▶

● 文字の間隔を変更する ………… p.897

SECTION-382

行間を変更する

ここでは、スタイルシートで行間を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>行間を変更</h1>
    <form><input type="button" value="CSSを変更" id="change"></form>
    <div>CSS and JavaScript<br>Sample code</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    var divElement = document.getElementsByTagName("div")[0];  // div要素を読み出し
    divElement.style.lineHeight = "50pt";  // 行間を設定
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、行間が変更されます。



ONEPOINT 行間を変更するにはスタイルシートの「lineHeight」プロパティを使う

行間を変更するには、スタイルシートの「lineHeight」プロパティに値を設定します。設定する値は、単位付きの文字列になります。たとえば、「"20pt"」とすると、行と行の間隔は20ptになります。行間は「180%」など、CSSで指定できる形式になります。

行揃えを変更する

ここでは、スタイルシートで行揃えを変更する方法について解説します。

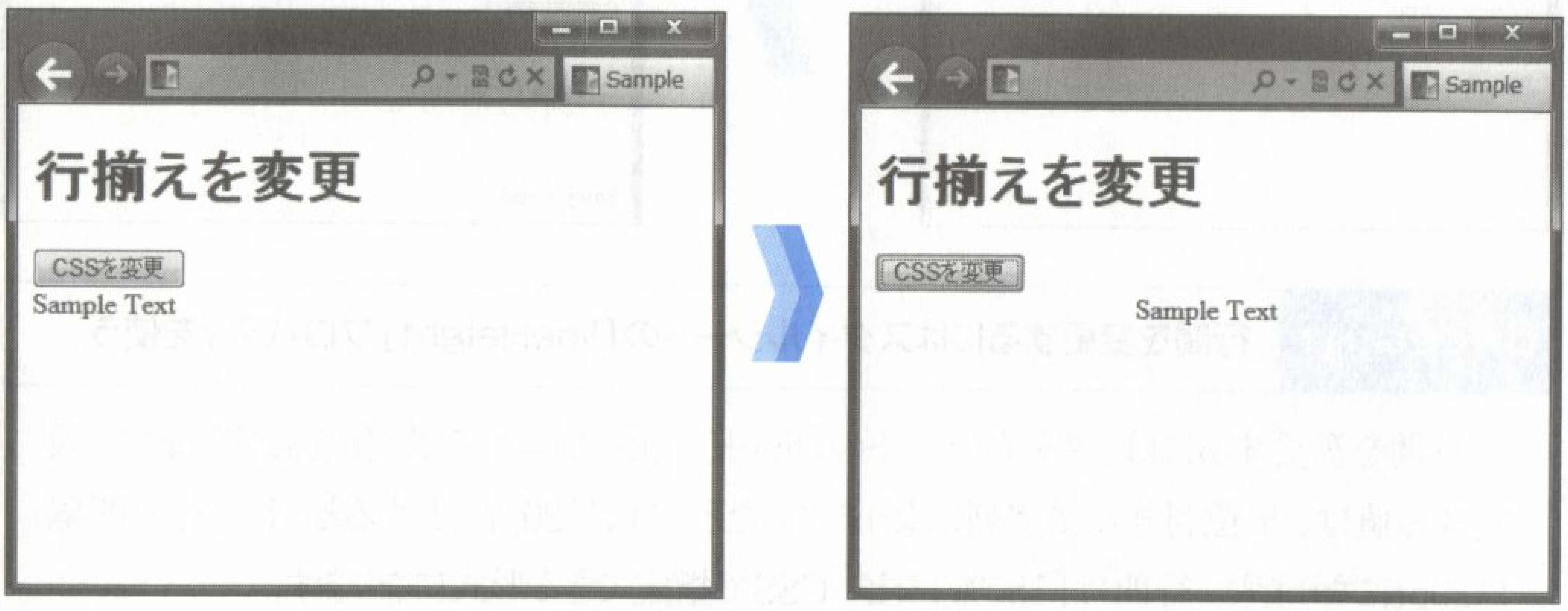
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>行揃えを変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 行揃えを設定
    divElement.style.textAlign = "center";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、行揃えが変更されます。

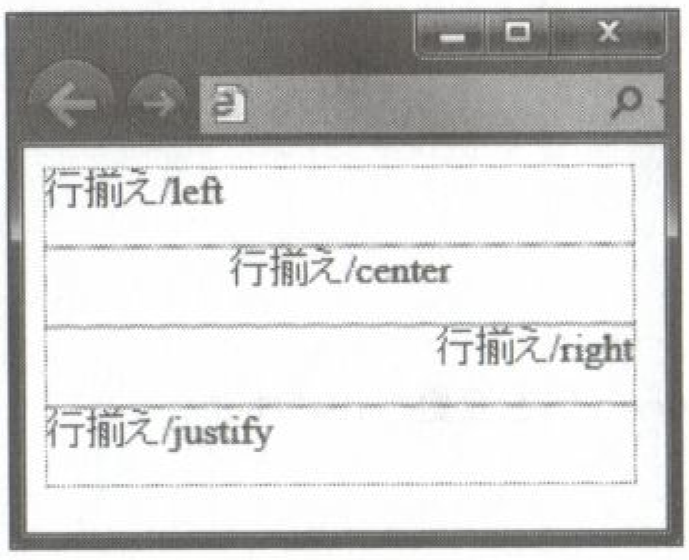


ONEPOINT 行揃えを変更するにはスタイルシートの「textAlign」プロパティを使う

行揃えを変更するには、スタイルシートの「textAlign」プロパティに行揃えを示す文字列を指定します。指定できる文字列は、次の表のようになります。

文字列	説明
left	左揃え
center	中央揃え（センタリング）
right	右揃え
justify	均等割り付け

それぞれの文字列を指定した結果は、次のようになります。



IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-384

文字を装飾する

ここでは、スタイルシートで文字を装飾する方法について解説します。

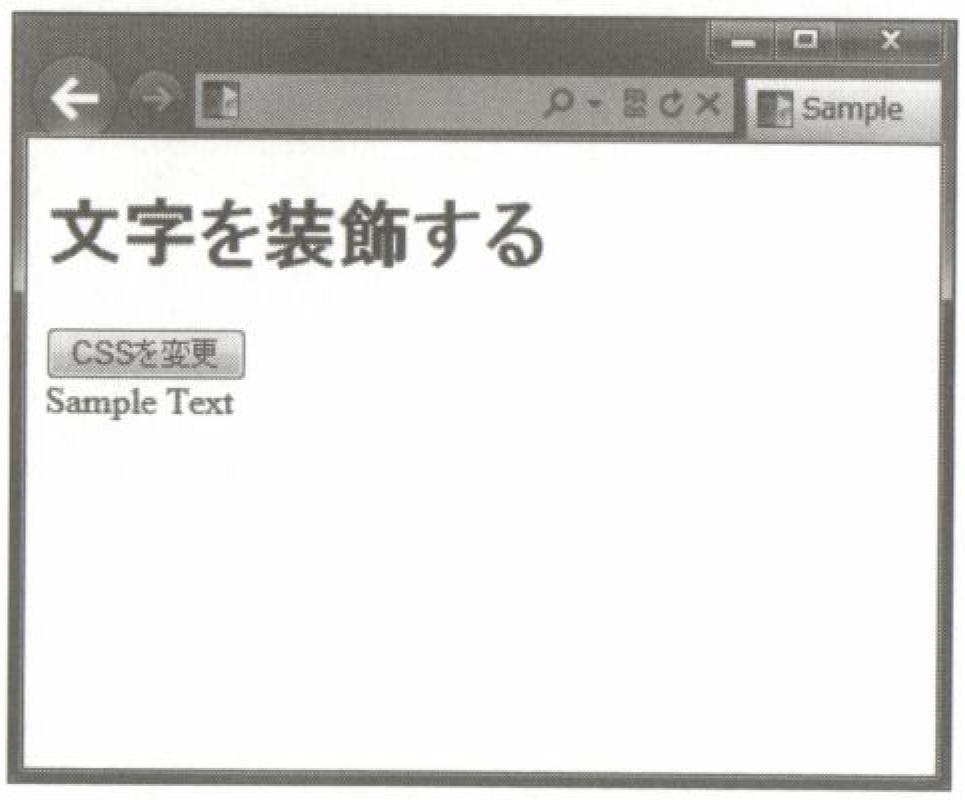
SAMPLE CODE HTMLのコード(index.html)

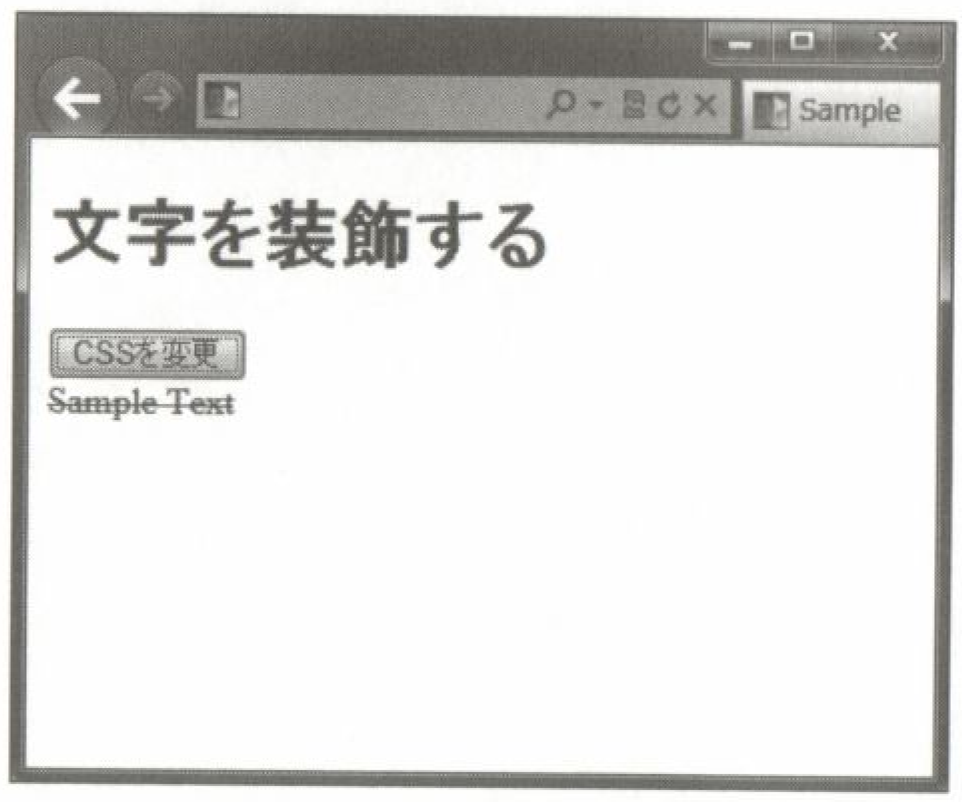
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字を装飾する</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 文字に取り消し線を設定
    divElement.style.textDecoration = "line-through";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字に取り消し線が表示されます。



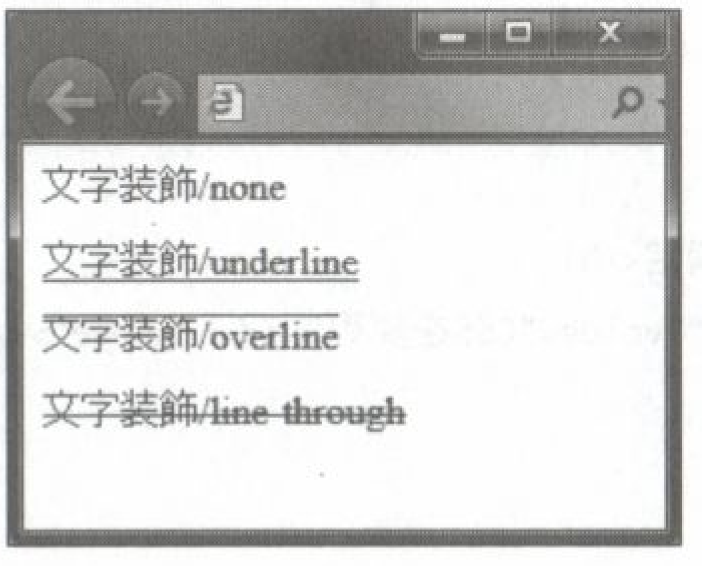


ONEPOINT 文字を装飾するには スタイルシートの「textDecoration」プロパティを使う

文字を装飾するにはスタイルシートの「textDecoration」プロパティに装飾するための文字列を指定します。指定できる文字列は、次の表のようになります。

文字列	説明
none	なし（通常の状態）
underline	下線（アンダーライン）
overline	上線（オーバーライン）
line-through	取り消し線

それぞれの文字列を指定した結果は、次のようになります。



字下げ(インデント)を変更する

ここでは、スタイルシートで字下げ(インデント)を変更する方法について解説します。

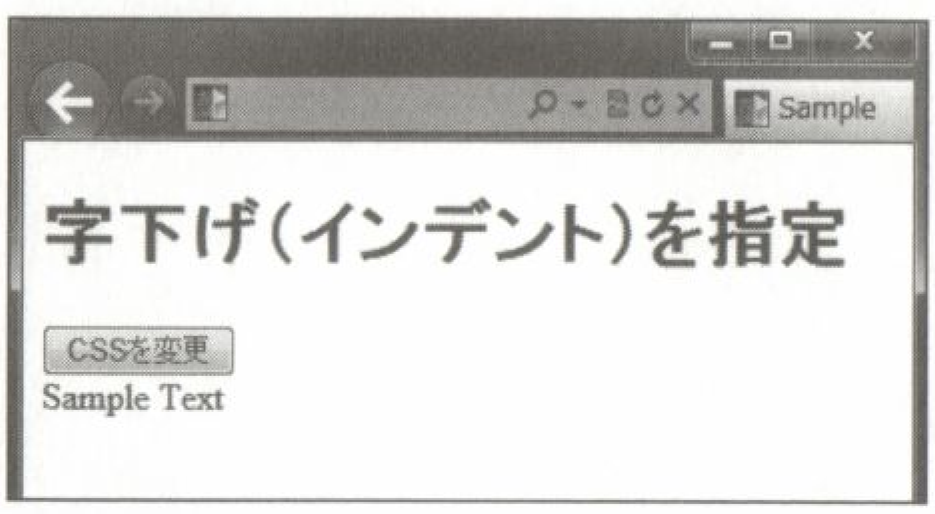
SAMPLE CODE HTMLのコード(index.html)

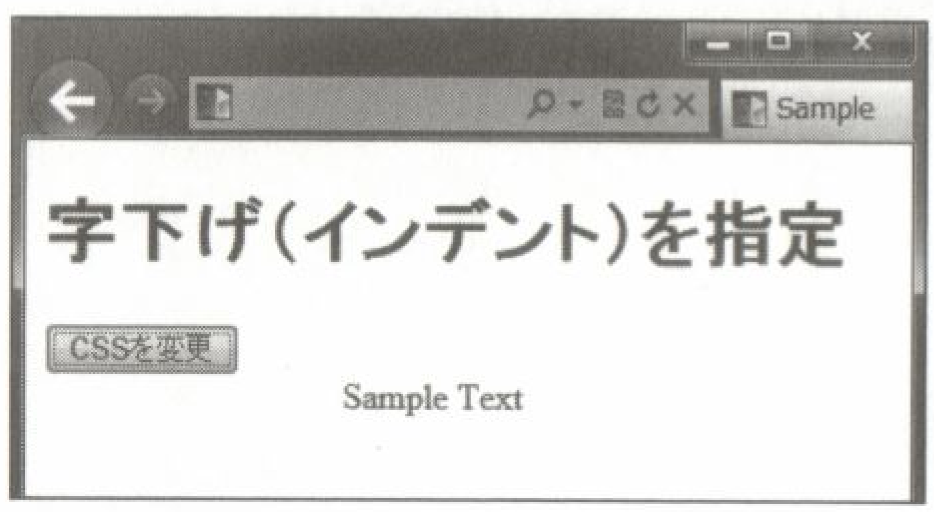
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>字下げ(インデント)を指定</h1>
    <form><input type="button" value="CSSを変更" id="change"></form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    var divElement = document.getElementsByTagName("div")[0];  // div要素を読み出し
    divElement.style.textIndent = "100pt";  // 字下げ(インデント)を設定
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、最初の行の字下げ(インデント)が変更されます。





ONEPOINT 字下げ(インデント)を変更するには
スタイルシートの「textIndent」プロパティを使う

字下げ(インデント)を変更するには、スタイルシートの「textIndent」プロパティに装飾するための値を指定します。指定する値は、単位付きの文字列になります。たとえば、「"50pt"」とすると、最初の行頭が筆記方向に50ptずれた位置から表示されるようになります。

英文字の表示形式を指定する

ここでは、スタイルシートで英文字の表示形式を指定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>英文字の表示方法を指定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>Sample Text</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 大文字に設定
    divElement.style.textTransform = "uppercase";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、英文字が大文字に変更されます。

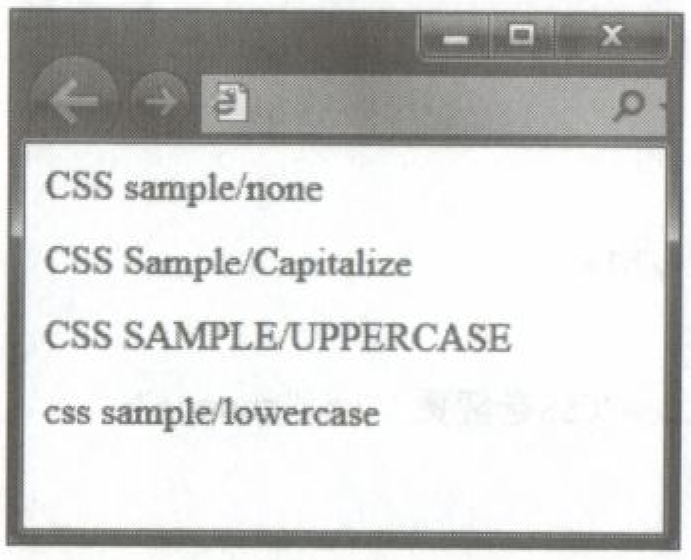


ONEPOINT 英文字の表示形式を指定するには スタイルシートの「textTransform」プロパティを使う

英文字の表示形式を指定するには、スタイルシートの「textTransform」プロパティに、次の表に示す文字列を設定します。

文字列	説明
none	通常の文字にする
capitalize	英文字の先頭文字を大文字にする
uppercase	英文字を大文字にする
lowercase	英文字を小文字にする

それぞれの文字列を指定した結果は、次のようになります。



関連項目 ▶▶▶

● 英文字の大文字/小文字に変換する ………… p.208
● 文字の形状を変更する ………… p.895

文字に影を表示する

ここでは、スタイルシートで文字に影を表示する方法について解説します。

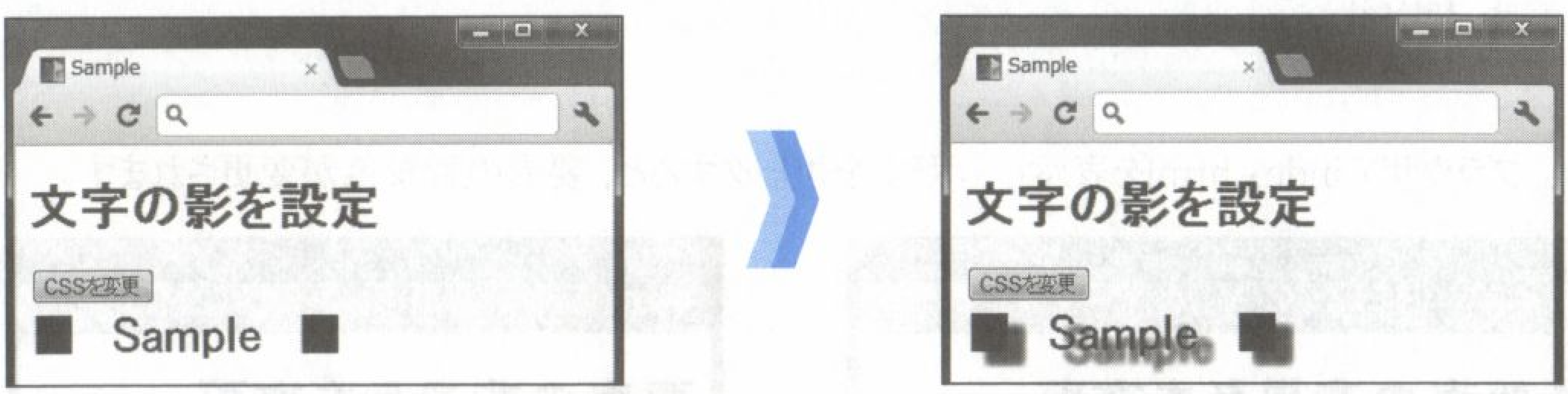
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>文字の影を設定</h1>
    <form><input type="button" value="CSSを変更" id="change"></form>
    <div style="font-size:24pt;">■  Sample  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    var divElement = document.getElementsByTagName("div")[0]; // div要素を読み出し
    divElement.style.textShadow = "10px 8px 5px blue"; // 文字の影を設定
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、文字に青い影が表示されます。



ONEPOINT 文字に影を表示するには スタイルシートの「textShadow」プロパティを使う

文字に影を表示するには、スタイルシートの「textShadow」プロパティに影を示す値を設定します。複数の値をまとめて指定します。その際、「影の横のずれ具合」「影の縦のずれ具合」「影のぼかし幅」「色」の順番で影の位置や色などを指定し、「textShadow」プロパティに文字列として一括で設定します。

IE 6 | Chrome | Safari 5 | Firefox 4 | Opera 11 | iOS 4 | Android 2 | WP

SECTION-388

要素の背景色を変更する

ここでは、スタイルシートで要素の背景色を変更する方法について解説します。

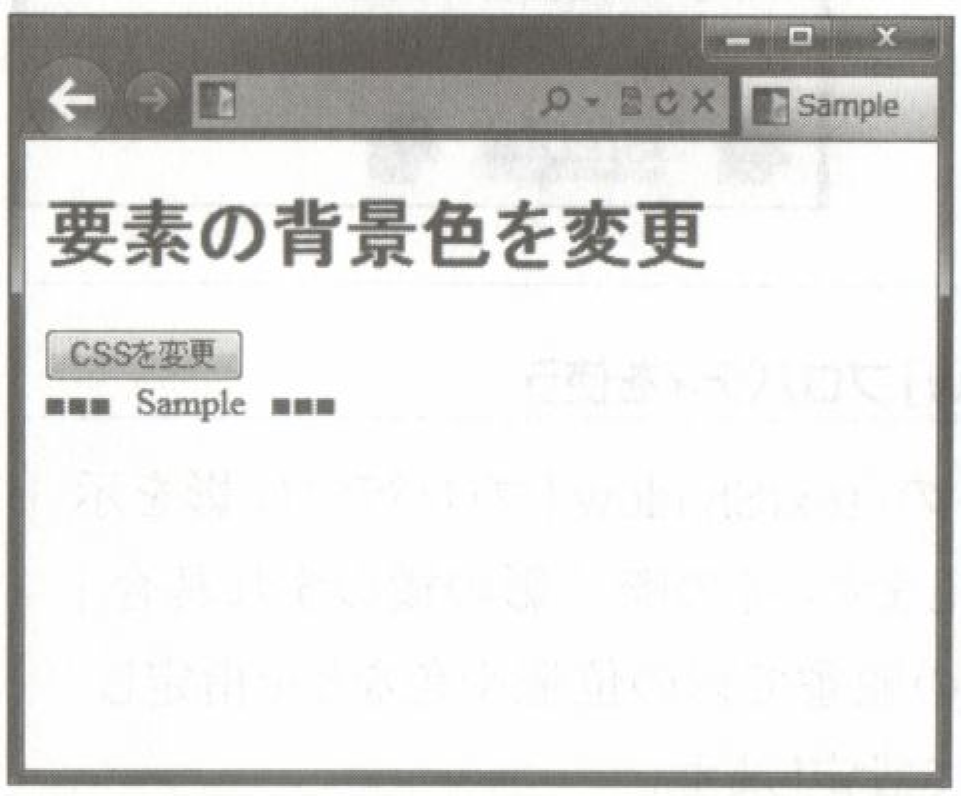
SAMPLE CODE HTMLのコード(index.html)

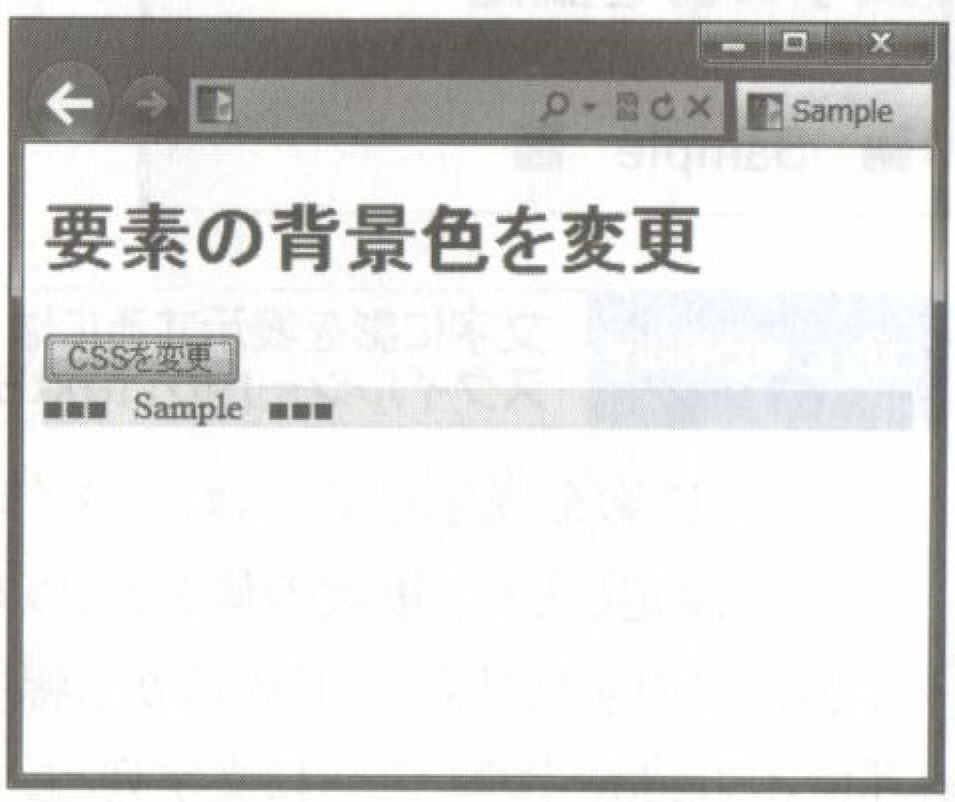
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の背景色を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■■■  Sample  ■■■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 背景色を設定
    divElement.style.backgroundColor = "yellow";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の背景色が変更されます。





ONEPOINT 要素の背景色を変更するにはスタイルシートの「color」プロパティを使う

要素の背景色を変更するには、スタイルシートの「color」プロパティに色を示すカラー名やカラーコードを指定します。カラー名は「red」や「blue」などの文字列を指定できます。また、RGBでの指定やHSL指定も可能です。指定可能なカラー形式は、次の表のようになります。

形式	説明
カラー名	「red」や「green」など、あらかじめ定義されている文字列
#rrggbb	16進数6桁でRGBの輝度を指定(00～ff)
#rgb	16進数3桁でRGBの輝度を指定(0～f)
rgb(赤, 緑, 青)	0～255までの輝度を指定。%での指定も可能
rgba(赤, 緑, 青, 不透明度)	0～255までの輝度と、0.0～1.0までの不透明度を指定
hsl(色相, 彩度, 輝度)	色相(0～360)、彩度(%)、輝度(%)を指定
hsla(色相, 彩度, 輝度, 不透明度)	色相(0～360)、彩度(%)、輝度(%)と0.0～1.0までの不透明度を指定
transparent	透明。「transparent」の文字をそのまま指定

詳しくは、次のURLを確認してください。

- CSS3 Color

URL http://www.w3.org/TR/css3-color/

関連項目 ▶▶▶

● 要素の背景画像を変更する ………………………………………… p.912

SECTION-389

要素の背景画像を変更する

ここでは、スタイルシートで要素の背景画像を変更する方法について解説します。

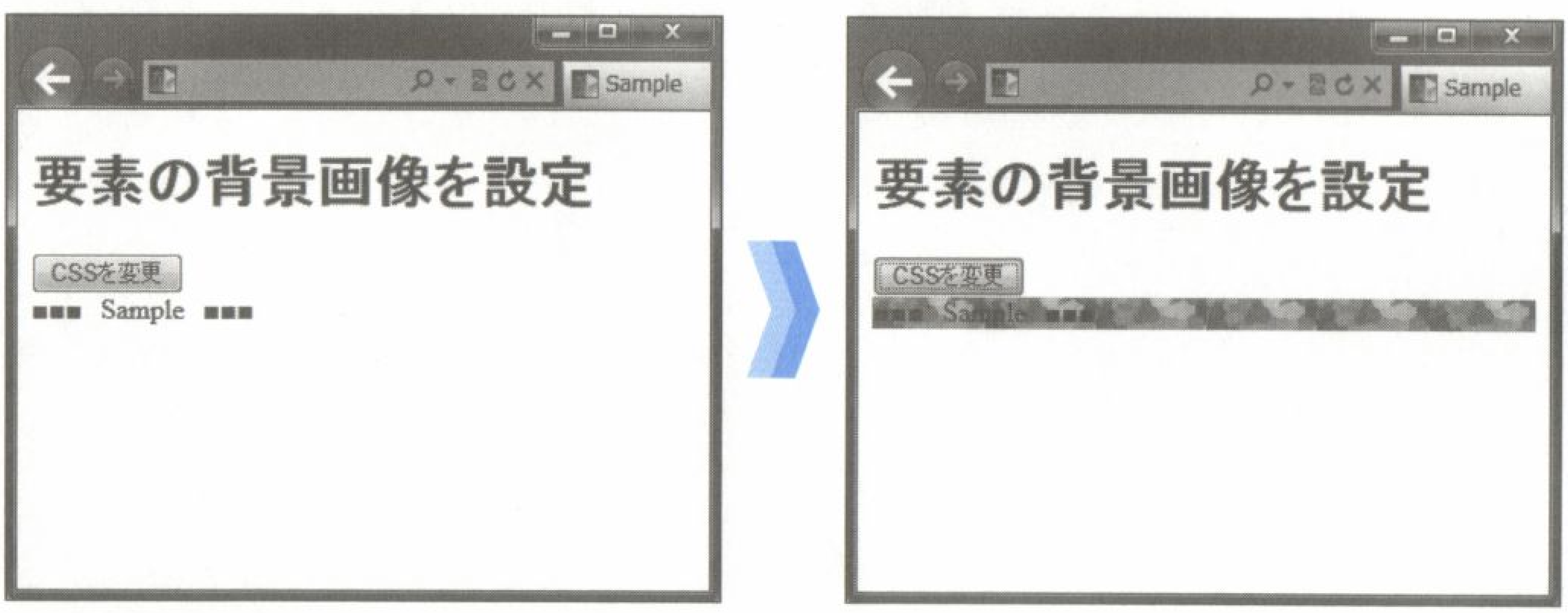
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の背景画像を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■■■  Sample  ■■■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 背景画像を設定
    divElement.style.backgroundImage = "url(images/bg.jpg)";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の背景画像が変更されます。



ONEPOINT **要素の背景画像を変更するには スタイルシートの「backgroundImage」プロパティを使う**

要素の背景画像を変更するには、スタイルシートの「backgroundImage」プロパティに背景画像があるURLを指定します。URLの指定は、「url(画像のあるURL)」として文字列で指定します。その際、画像の基準パスはHTMLファイルがあるディレクトリ(フォルダ)になります。

関連項目 ▶▶▶

- 要素の背景色を変更する …… p.910
- 要素の背景画像の位置を変更する …… p.914
- 要素の背景画像の繰り返し方法を変更する …… p.916
- 背景画像の固定方法を変更する …… p.918

要素の背景画像の位置を変更する

ここでは、スタイルシートで要素の背景画像の位置を変更する方法について解説します。

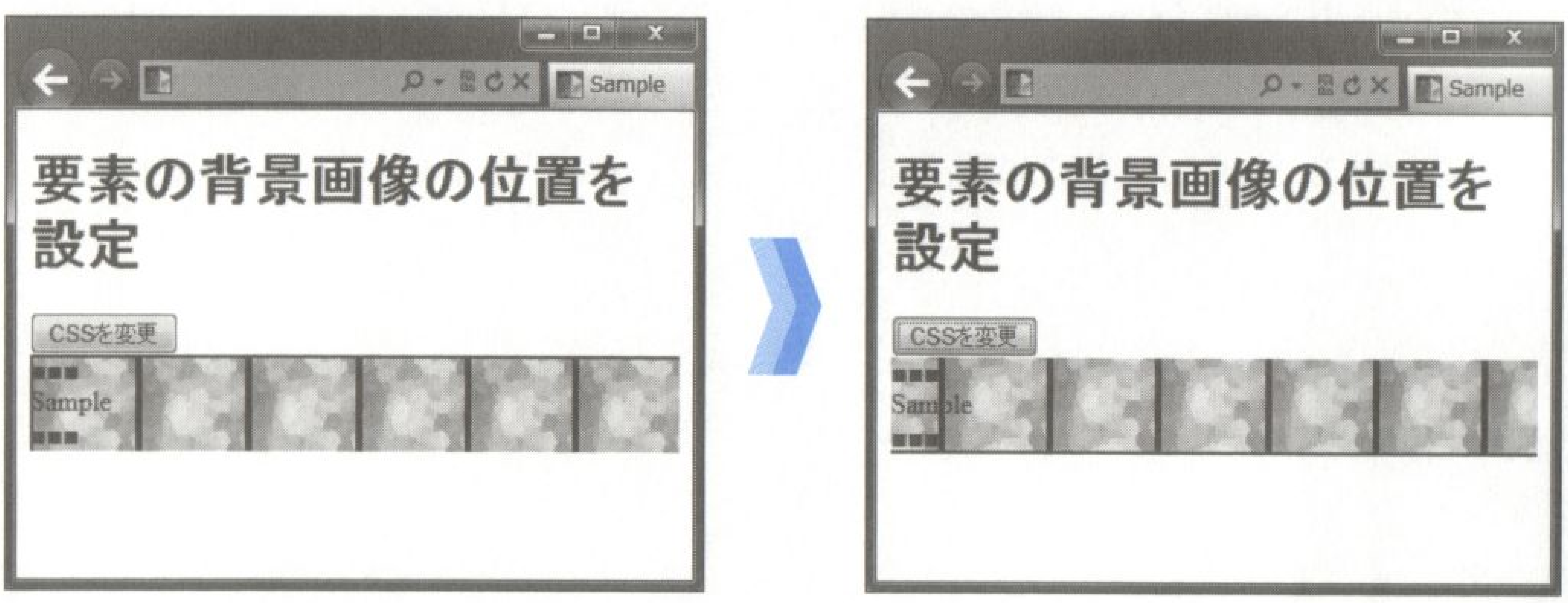
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の背景画像の位置を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div style="background-image:url(images/bg.jpg)">■■■<br>Sample<br>■■■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 背景画像の位置を設定
    divElement.style.backgroundPosition = "30px bottom";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の背景画像の位置が変更されます。



ONEPOINT **要素の背景画像の位置を変更するには スタイルシートの「backgroundPosition」プロパティを使う**

要素の背景画像の位置を変更するには、スタイルシートの「backgroundPosition」プロパティに横方向と縦方向の位置を指定します。位置は、文字列でまとめてX座標とY座標を指定します。たとえば、「"10px 5mm"」とすると、背景画像の基準表示位置が横に10px、縦方向に5mm移動します。また、次の表に示す文字列で位置を指定することもできます。

文字列	説明
left	背景画像の左側
center	背景画像の中央(縦横共通)
right	背景画像の右側
top	背景画像の上側
bottom	背景画像の下側

関連項目 ▶▶▶

● 要素の背景画像を変更する ………… p.912
● 要素の背景画像の繰り返し方法を変更する ………… p.916
● 背景画像の固定方法を変更する ………… p.918

要素の背景画像の繰り返し方法を変更する

ここでは、スタイルシートで要素の背景画像の繰り返し方法を変更する方法について解説します。

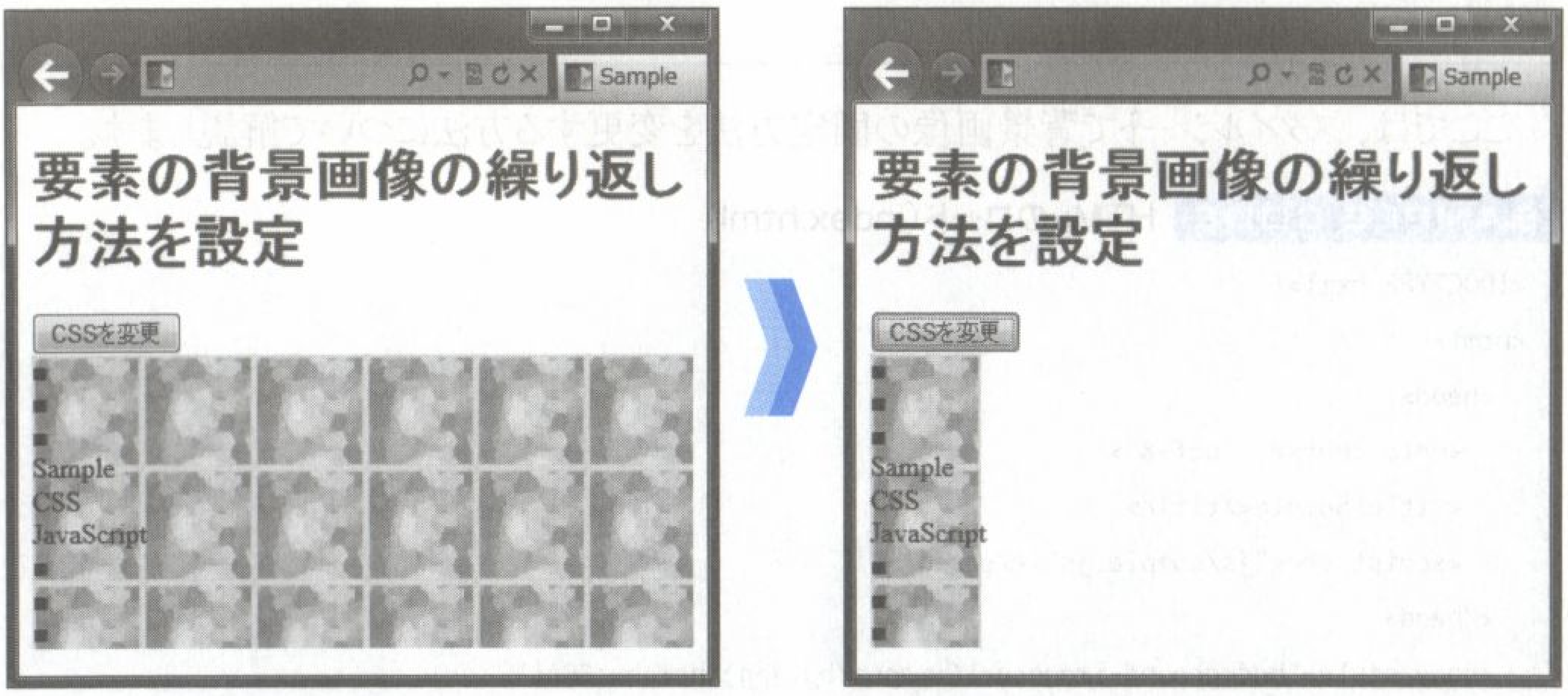
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の背景画像の繰り返し方法を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div style="background-image:url(images/bg.jpg)">
    ■<br>■<br>■<br>Sample<br>CSS<br>JavaScript<br>■<br>■<br>■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 背景画像の繰り返し方法を設定
    divElement.style.backgroundRepeat = "repeat-y";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の背景画像が縦方向だけに繰り返されます。



ONEPOINT **要素の背景画像の繰り返し方法を変更するには スタイルシートの「backgroundRepeat」プロパティを使う**

要素の背景画像の繰り返し方法を変更するには、スタイルシートの「background Repeat」プロパティに、次の表に示す文字列を指定します。

文字列	説明
no-repeat	繰り返さない
repeat-x	横方向だけに繰り返し表示する
repeat-y	横方向だけに繰り返し表示する
repeat	横方向と縦方向に繰り返し表示する(デフォルト)

関連項目 ▶▶▶

- 要素の背景画像を変更する ……………… p.912
- 要素の背景画像の位置を変更する ……………… p.914
- 背景画像の固定方法を変更する ……………… p.918

SECTION-392

背景画像の固定方法を変更する

ここでは、スタイルシートで背景画像の固定方法を変更する方法について解説します。

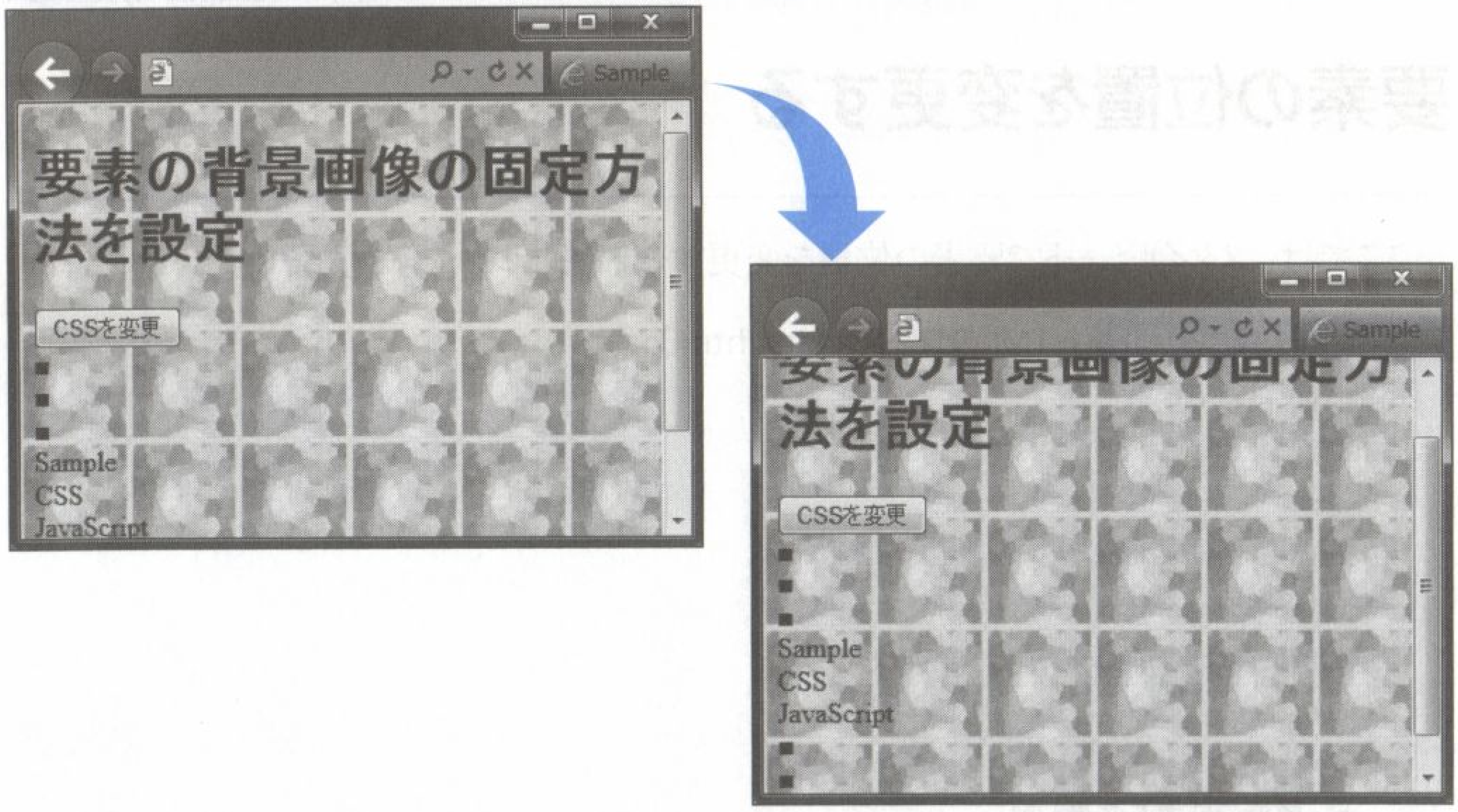
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body style="background-image:url(images/bg.jpg);height:80pt">
    <h1>要素の背景画像の固定方法を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>
    ■<br>■<br>■<br>Sample<br>CSS<br>JavaScript<br>■<br>■<br>■</div>
  </body>
</html>
```

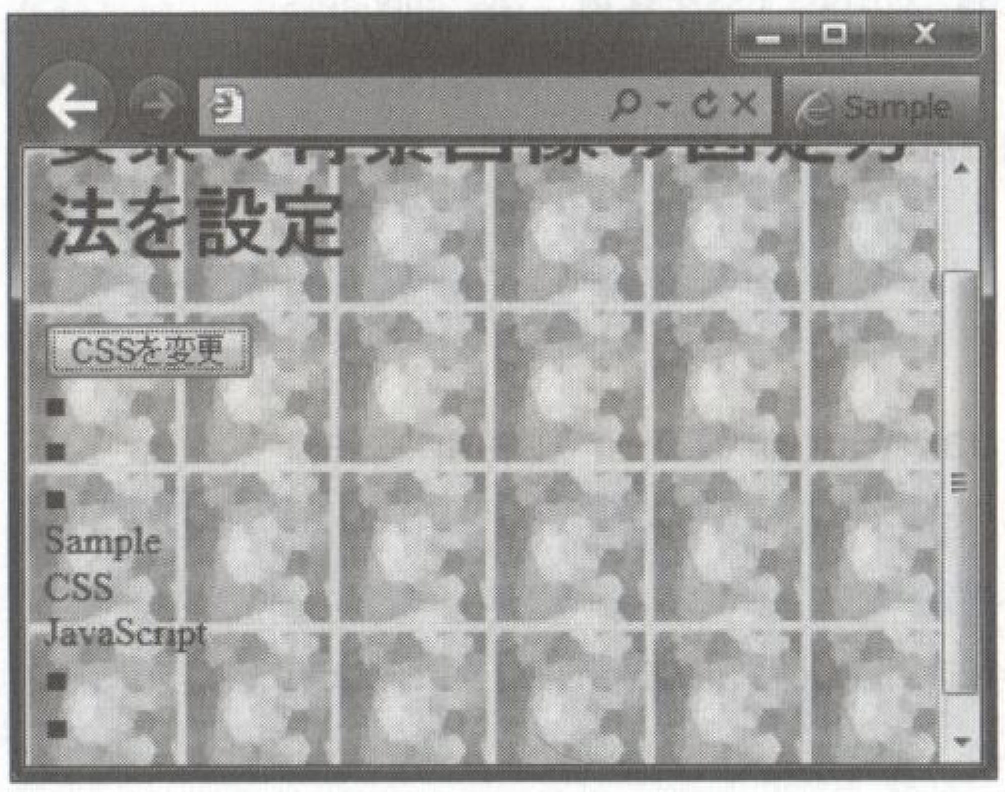
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.body;
    // 背景画像の固定方法を設定
    divElement.style.backgroundAttachment = "fixed";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示します。背景が固定されていない状態では、本文に合わせて背景画像もスクロールします。



ボタンをクリックすると、背景画像が固定され、本文のみがスクロールします。



ONEPOINT 要素の背景画像の固定方法を変更するにはスタイルシートの「backgroundAttachment」プロパティを使う

要素の背景画像の固定方法を変更するには、スタイルシートの「background Attachment」プロパティに、次の表に示す文字列を指定します。文字列は、「"fixed"」のように、文字列で指定する必要があります。

文字列	説明
scroll	本文にあわせて背景もスクロールする（デフォルト）
fixed	背景画像を固定する。本文のみスクロールする

なお、Windows Phoneでは「fixed」に設定しても背景は固定されず、常にスクロールする状態になります。

関連項目 ▶▶▶

- 要素の背景画像を変更する …… p.912
- 要素の背景画像の位置を変更する …… p.914
- 要素の背景画像の繰り返し方法を変更する …… p.916

SECTION-393

要素の位置を変更する

ここでは、スタイルシートで要素の位置を変更する方法について解説します。

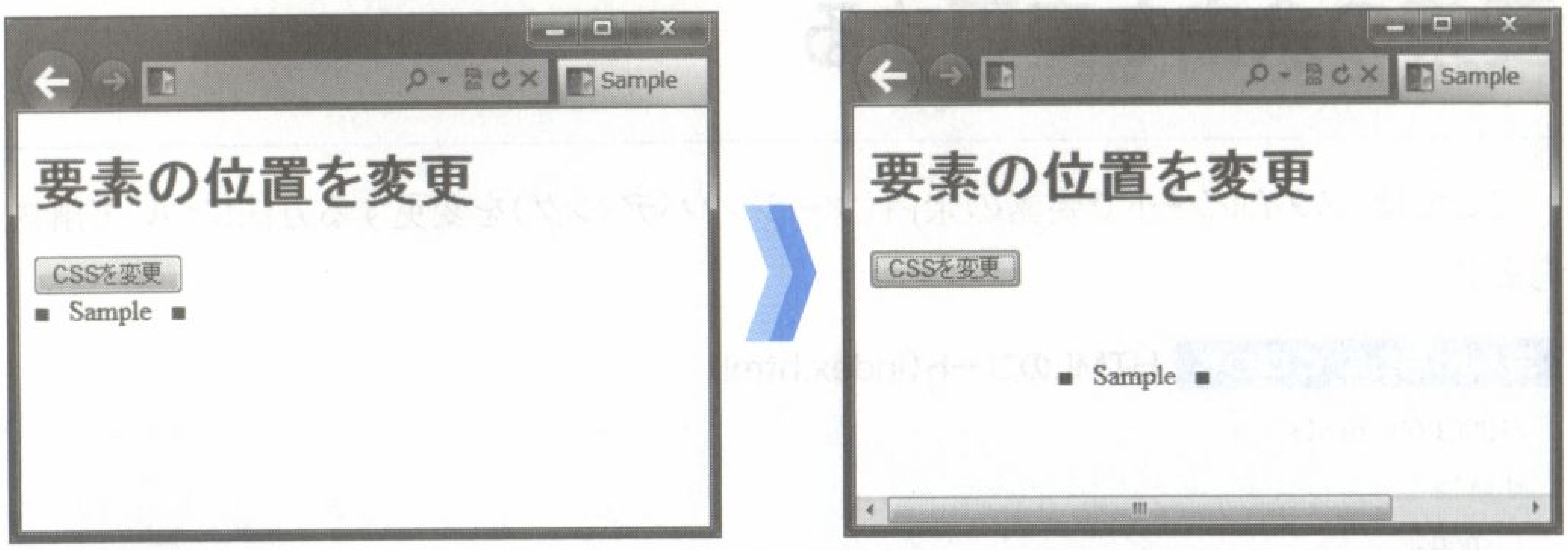
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の位置を変更</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■  Sample  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 座標系を設定
    divElement.style.position = "relative";
    // X座標を設定
    divElement.style.left = "80pt";
    // Y座標を設定
    divElement.style.top = "30pt";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の位置が変更されます。



ONEPOINT

要素の位置を変更するには スタイルシートの「position」プロパティを使う

要素の位置を変更するには、スタイルシートの「position」プロパティに、座標系を示す文字列を設定します。すでにCSSで設定されている場合は、既存の座標系が利用されます。座標系を示す文字列は、次の表のようになります。

文字列	説明
absolute	絶対座標系
relative	相対座標系
static	指定なし(デフォルト)
fixed	絶対座標系(左上に固定)

要素のX座標は、スタイルシートの「left」プロパティに単位付きで設定します。Y座標も同様に、「top」プロパティに設定します。なお、座標の原点は常に左上になり、右下にいくに従って座標値が大きくなります。

SECTION-394

要素の余白を変更する

ここでは、スタイルシートで要素の余白(マージン/パディング)を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の余白を変更</h1>
    <form>
      <input type="button" value="marginを変更" id="change1">
      <input type="button" value="paddingを変更" id="change2">
    </form>
    <div style="border:1px solid black">■ Sample ■</div>
  </body>
</html>
```

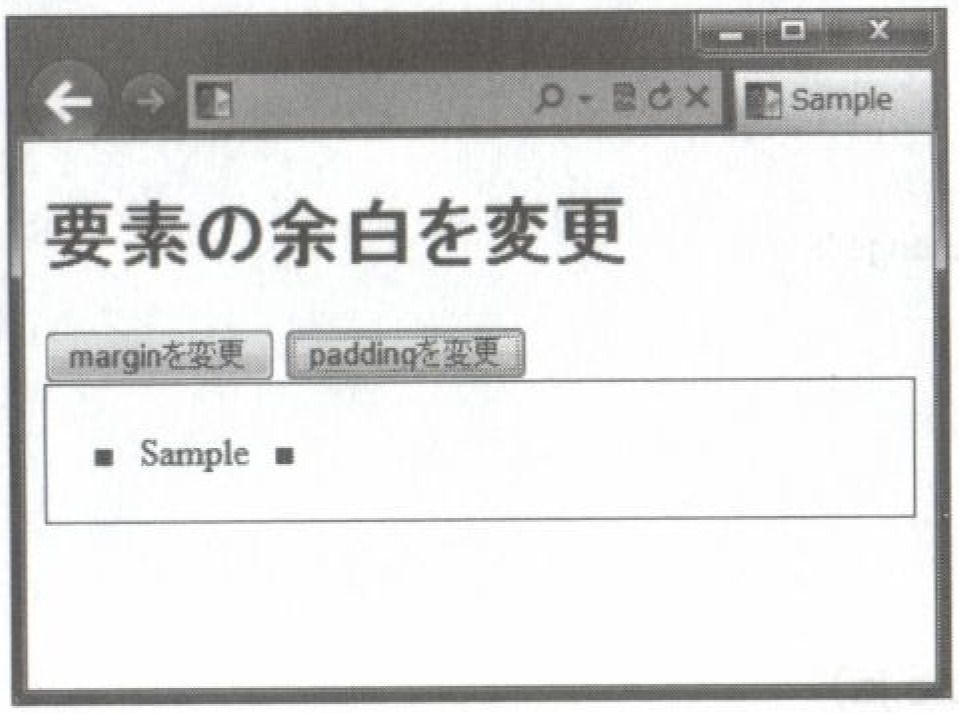
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change1").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // マージン(余白)を設定
    divElement.style.margin = "20px";
  }, false);
  // ボタンにイベントを設定
  document.getElementById("change2").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // パディング(余白)を設定
    divElement.style.padding = "20px";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、「marginを変更」ボタンをクリックすると、要素のマージン(枠の外側)が変更されます。



また、「paddingを変更」ボタンをクリックすると、要素のマージン(枠の内側)が変更されます。



ONEPOINT　要素の余白を変更するには スタイルシートの「margin」プロパティや「padding」プロパティを使う

要素の余白を変更するには、スタイルシートの「margin」プロパティ、または、「padding」プロパティに値を指定します。「margin」プロパティは、要素の外側の余白を指定します。指定できる値は、CSSで指定できる形式になります。「"10pt"」と指定すると、上下左右の余白が10ptになります。「"10pt 20pt 30pt 40pt"」とすると、上側に10pt、右側に20pt、下側に30pt、左側に40ptの余白が設定されます。

要素の内側の余白を指定するには、「padding」プロパティに値を指定します。指定できる値は、「margin」プロパティと同様に、CSSで指定できる形式になります。

SECTION-395

要素の枠を変更する

ここでは、スタイルシートで要素の枠を変更する方法について解説します。

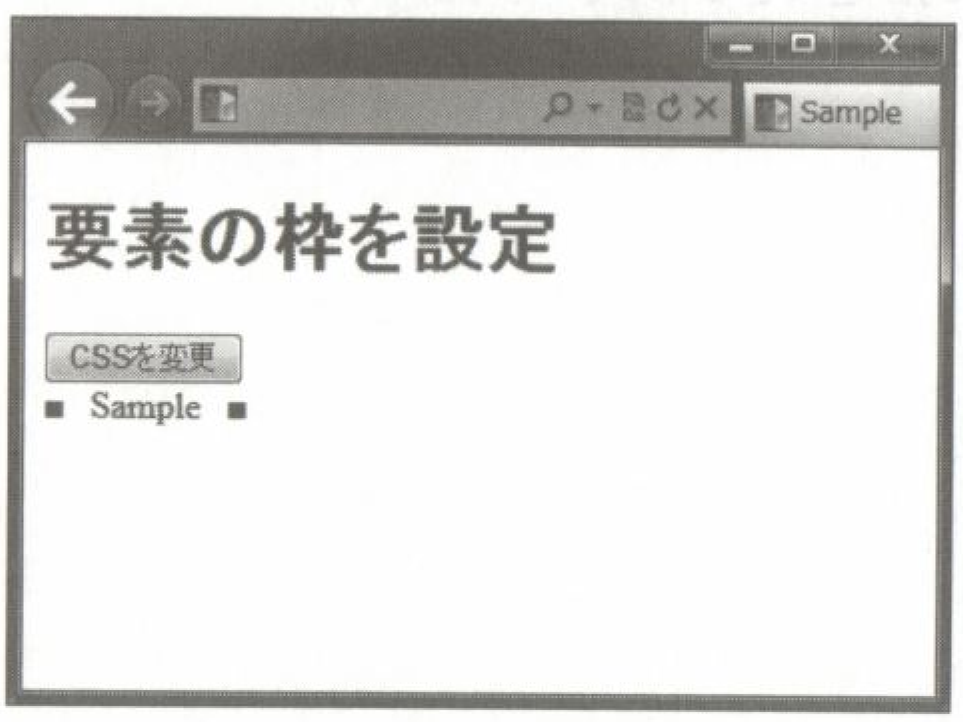
SAMPLE CODE HTMLのコード(index.html)

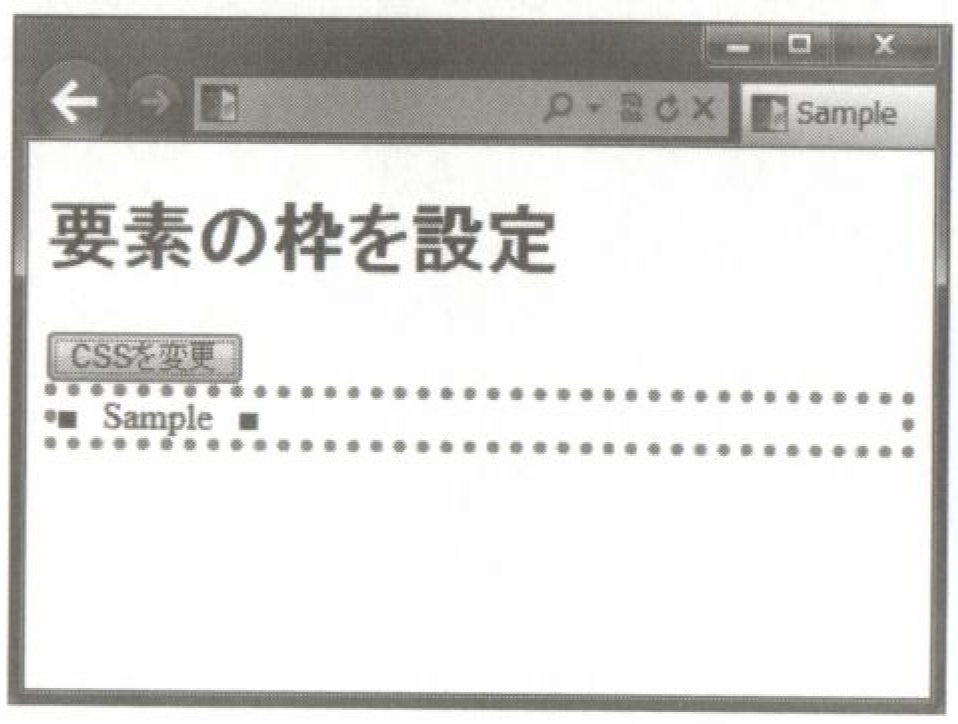
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の枠を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■ Sample ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 枠のスタイルなどを設定
    divElement.style.border = "5px dotted red";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の枠が変更されます。





ONEPOINT 要素の枠を変更するにはスタイルシートの「border」プロパティを使う

要素の枠を変更するには、スタイルシートの「border」プロパティに枠に関する値を設定します。「border」プロパティには、一括して枠の値を指定することができます。たとえば、「"2px solid black"」とすると、2pxの黒い枠が設定されます。

また、次の表に示すプロパティに値を設定することで、枠の形状などを変更することができます。

プロパティ	説明
borderBottom	枠の下側のスタイルを一括で指定する
borderLeft	枠の左側のスタイルを一括で指定する
borderRight	枠の右側のスタイルを一括で指定する
borderTop	枠の上側のスタイルを一括で指定する
borderColor	上下左右の枠の色を一括で指定する
borderStyle	上下左右の枠の形状を一括で指定する
borderWidth	上下左右の枠の幅を一括で指定する
borderBottomColor	下側の枠の色を指定する
borderBottomStyle	下側の枠の形状を指定する
borderBottomWidth	下側の枠の幅を指定する
borderTopColor	上側の枠の色を指定する
borderTopStyle	上側の枠の形状を指定する
borderTopWidth	上側の枠の幅を指定する
borderLeftColor	左側の枠の色を指定する
borderLeftStyle	左側の枠の形状を指定する
borderLeftWidth	左側の枠の幅を指定する
borderRightColor	右側の枠の色を指定する
borderRightStyle	右側の枠の形状を指定する
borderRightWidth	右側の枠の幅を指定する

要素の表示形式を変更する

ここでは、スタイルシートで要素の表示形式を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の表示形式を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■　Sample1　■</div>
    <div>■　Sample2　■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の表示形式を設定
    divElement.style.display = "none";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の表示形式が変更されます。ここでは、最初の「div」要素が表示されなくなり、下の「div」要素が上にずれます。



ONEPOINT

要素の表示形式を変更するには スタイルシートの「display」プロパティを使う

要素の表示形式を変更するには、スタイルシートの「display」プロパティに次の表に示す値(文字列)を設定します。

値	説明
none	表示しない
block	ブロック要素として表示する
inline	インライン要素として表示する
list-item	リスト項目として表示する
marker	リスト項目のマーカー。CSS2.1では使用不可
compact	ブロック要素、または、インライン要素として表示する(「dt」要素)。CSS2.1では使用不可
run-in	ブロック要素またはインライン要素として表示する
table	テーブル要素として表示する
inline-table	テーブル要素として表示する(インライン要素)
table-caption	テーブルのキャプションとして表示する
table-row	テーブルの行として表示する
table-cell	テーブルのセルとして表示する
table-column	テーブルの列として表示する
table-row-group	テーブルの列グループとして表示する
table-column-group	テーブルの行グループとして表示する
table-header-group	テーブルのヘッダーグループとして表示する
table-footer-group	テーブルのフッターグループとして表示する

関連項目 ▶▶▶

● 要素の表示方法を変更する ………… p.934

IE 6 Chrome Safari 5 Firefox 4 Opera 11 iOS 4 Android 2 WP

SECTION-397

要素の横幅・縦幅を変更する

ここでは、スタイルシートで要素の横幅・縦幅を変更する方法について解説します。

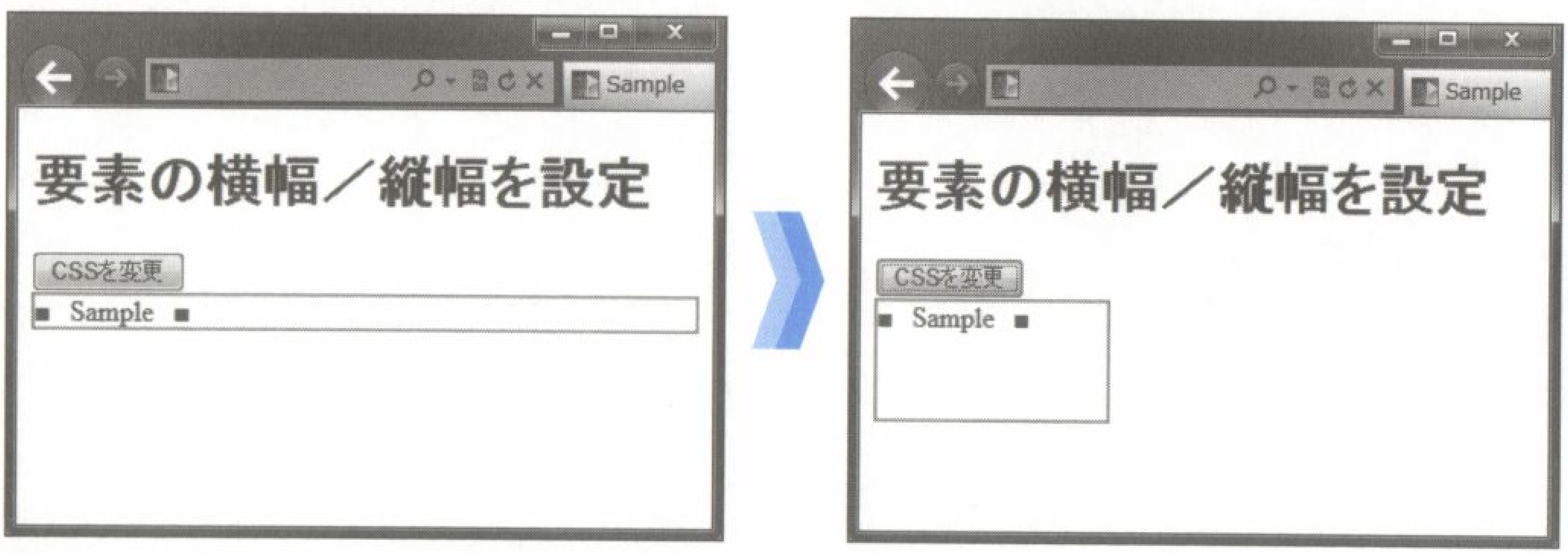
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の横幅／縦幅を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div style="border:1pt solid red">■ Sample ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の横幅を設定
    divElement.style.width = "100pt";
    // 要素の縦幅を設定
    divElement.style.height = "50pt";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の横幅･縦幅が変更されます。



ONEPOINT

要素の横幅を変更するにはスタイルシートの「width」プロパティ、縦幅を変更するにはスタイルシートの「height」プロパティを使う

要素の横幅を変更するには、スタイルシートの「width」プロパティにCSSで指定可能な値を設定します。たとえば、「"215pt"」とすると、横幅が215ptに設定されます。「"120mm"」なら、120mmに横幅が設定されます。

縦幅を変更するには、スタイルシートの「height」プロパティにCSSで指定可能な値を設定します。「"4cm"」とすると、縦幅は4cmに設定されます。

COLUMN

iPhone/Androidで表示領域のサイズを指定する

iPhone/Androidで画面の表示サイズを指定するには、「meta」要素を使います。幅は、「width」プロパティと「height」プロパティで指定します。

```
<meta name="viewport" content="width=320,height=480">
```

なお、「width」プロパティに「device-width」、「height」プロパティに「device-height」を指定すると、デバイスの横幅・縦幅になります。

SECTION-398

要素のはみ出した部分の表示方法を変更する

ここでは、スタイルシートで要素のはみ出した部分(オーバーフローした部分)の表示方法を変更する方法について解説します。

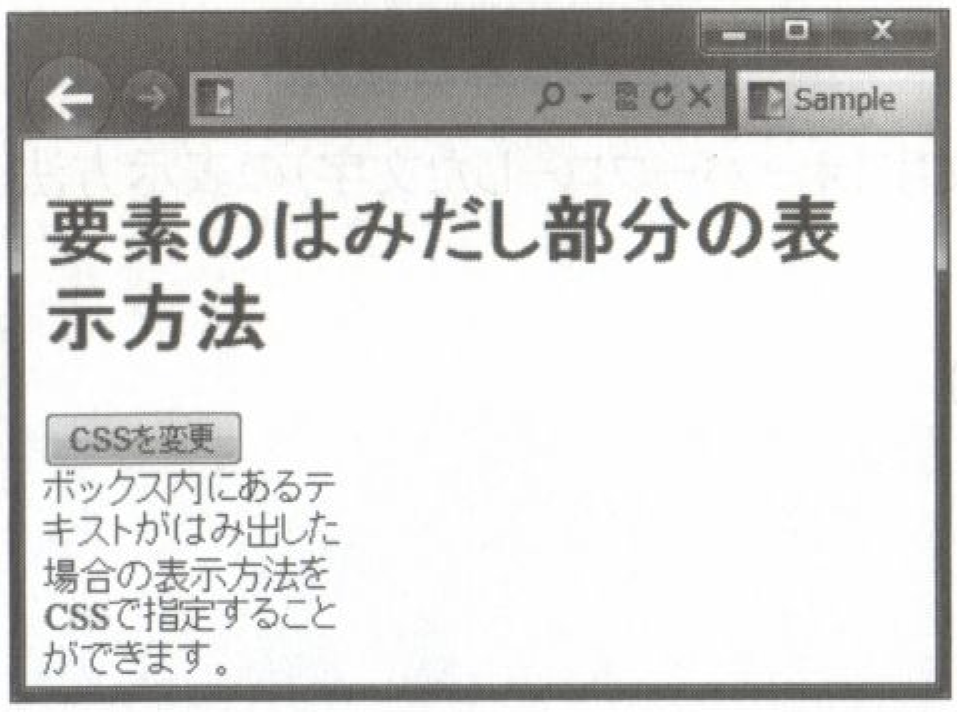
SAMPLE CODE HTMLのコード(index.html)

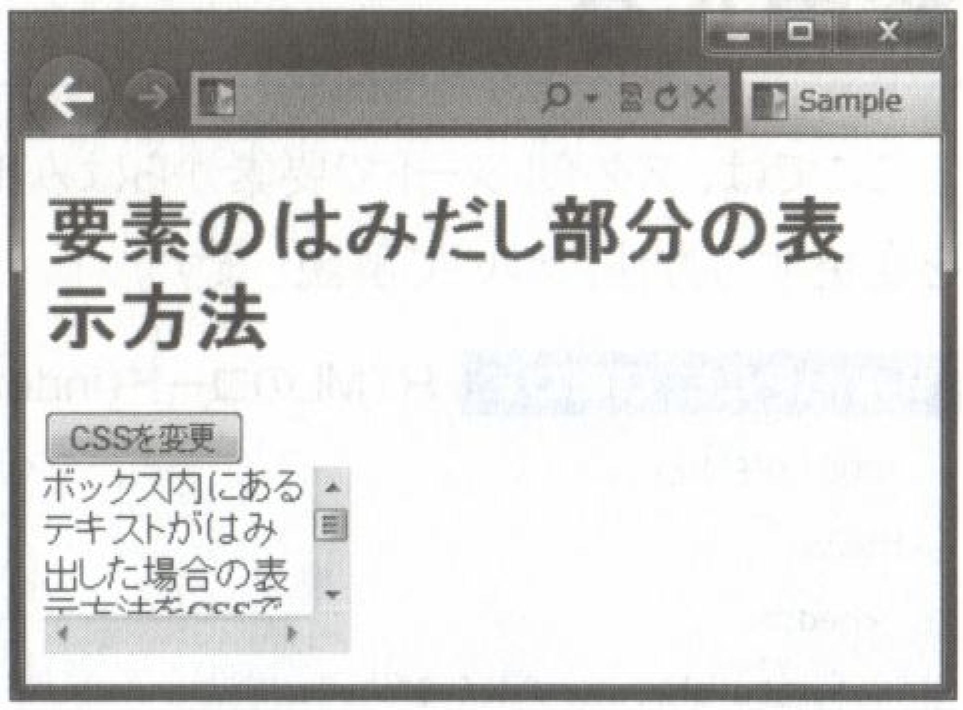
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素のはみだし部分の表示方法</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div style="width:100pt;height:60pt;">
    ボックス内にあるテキストがはみ出した場合の表示方法をCSSで指定することができます。
    </div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素のはみ出した部分の表示方法を設定
    divElement.style.overflow = "scroll";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素のはみ出した部分の表示方法が変更されスクロールバーが表示されます。





ONEPOINT 要素のはみ出した部分の表示方法を変更するには スタイルシートの「overflow」プロパティを使う

要素のはみ出した部分の表示方法を変更するには、スタイルシートの「overflow」プロパティに、次の表に示す文字列を指定します。領域の横幅は「width」プロパティ、縦幅は「height」プロパティで指定された値になります。

また、横方向だけを別途指定したい場合は「overflowX」プロパティ、縦方向だけを指定したい場合は「overflowY」プロパティに、次の表に示す文字列を指定します。

文字列	説明
visible	領域外も表示する
hidden	領域外は表示しない
scroll	領域からはみ出した場合はスクロールバーを表示する
auto	自動

関連項目 ▶▶▶

● 要素からはみ出した文字の表示方法を変更する ………………………………………… p.932

SECTION-399

要素からはみ出した文字の表示方法を変更する

ここでは、スタイルシートで要素からはみ出した文字(オーバーフローした文字)の表示方法を変更する方法について解説します。

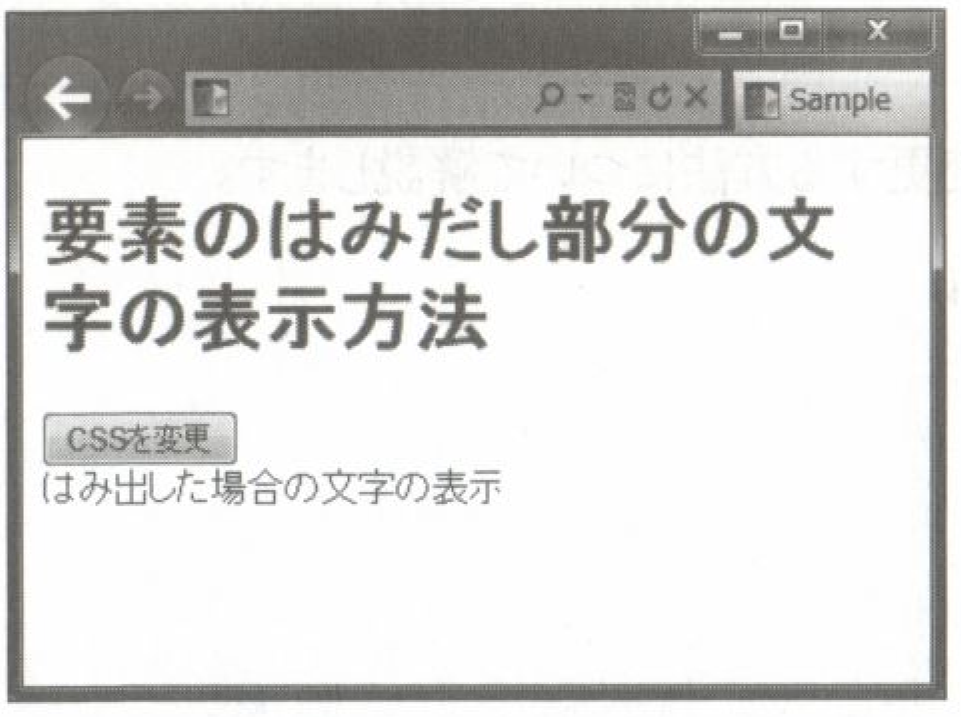
SAMPLE CODE HTMLのコード(index.html)

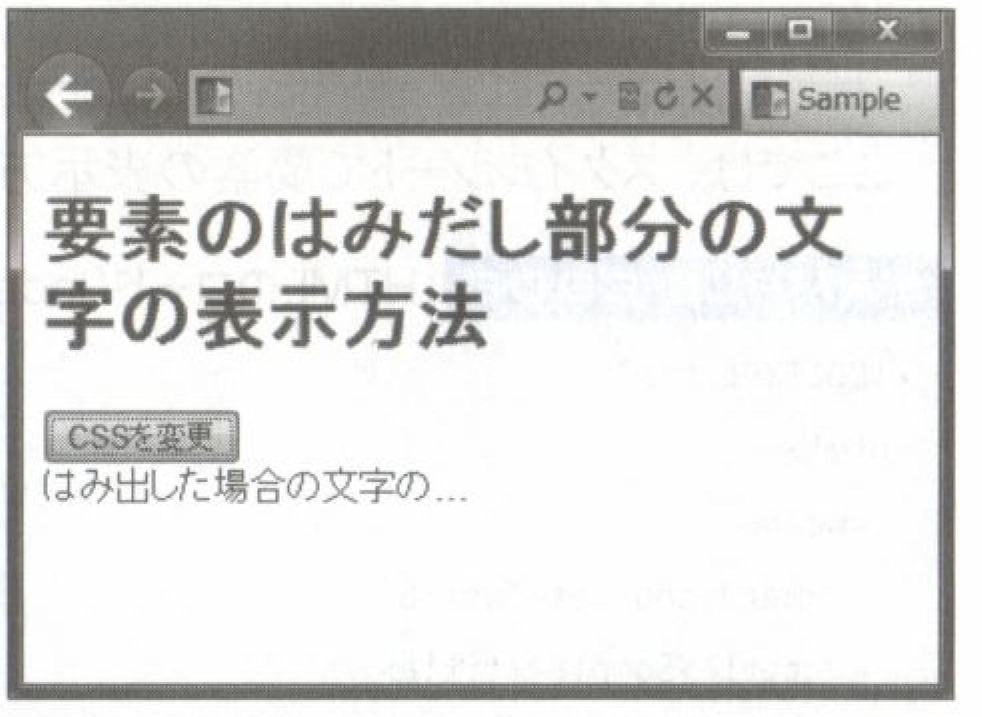
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素のはみだし部分の文字の表示方法</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div style="width:150pt;height:60pt;overflow:hidden">
      <nobr>はみ出した場合の文字の表示方法は？ </nobr>
    </div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素内の文字の末尾の表示方法を設定
    divElement.style.textOverflow = "ellipsis";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素のはみ出した部分の文字の表示方法が変更されます。





ONEPOINT **要素のはみだした部分の文字の表示方法を変更するには スタイルシートの「textOverflow」プロパティを使う**

要素のはみだした部分の文字の表示方法を変更するには、スタイルシートの「textOverflow」プロパティに、次の表に示す文字列を指定します。領域の横幅は「width」プロパティ、縦幅は「height」プロパティで指定された値になります。また、「textOverflow」プロパティに設定した値は、スタイルシートで「overflow:hidden」が設定されていないと反映されません。

文字列	説明
clip	領域外は表示しない
ellipsis	領域内の最後の文字を「…」にする

関連項目 ▶▶▶

● 要素のはみ出した部分の表示方法を変更する ……………… p.930

SECTION-400

要素の表示方法を変更する

ここでは、スタイルシートで要素の表示方法を変更する方法について解説します。

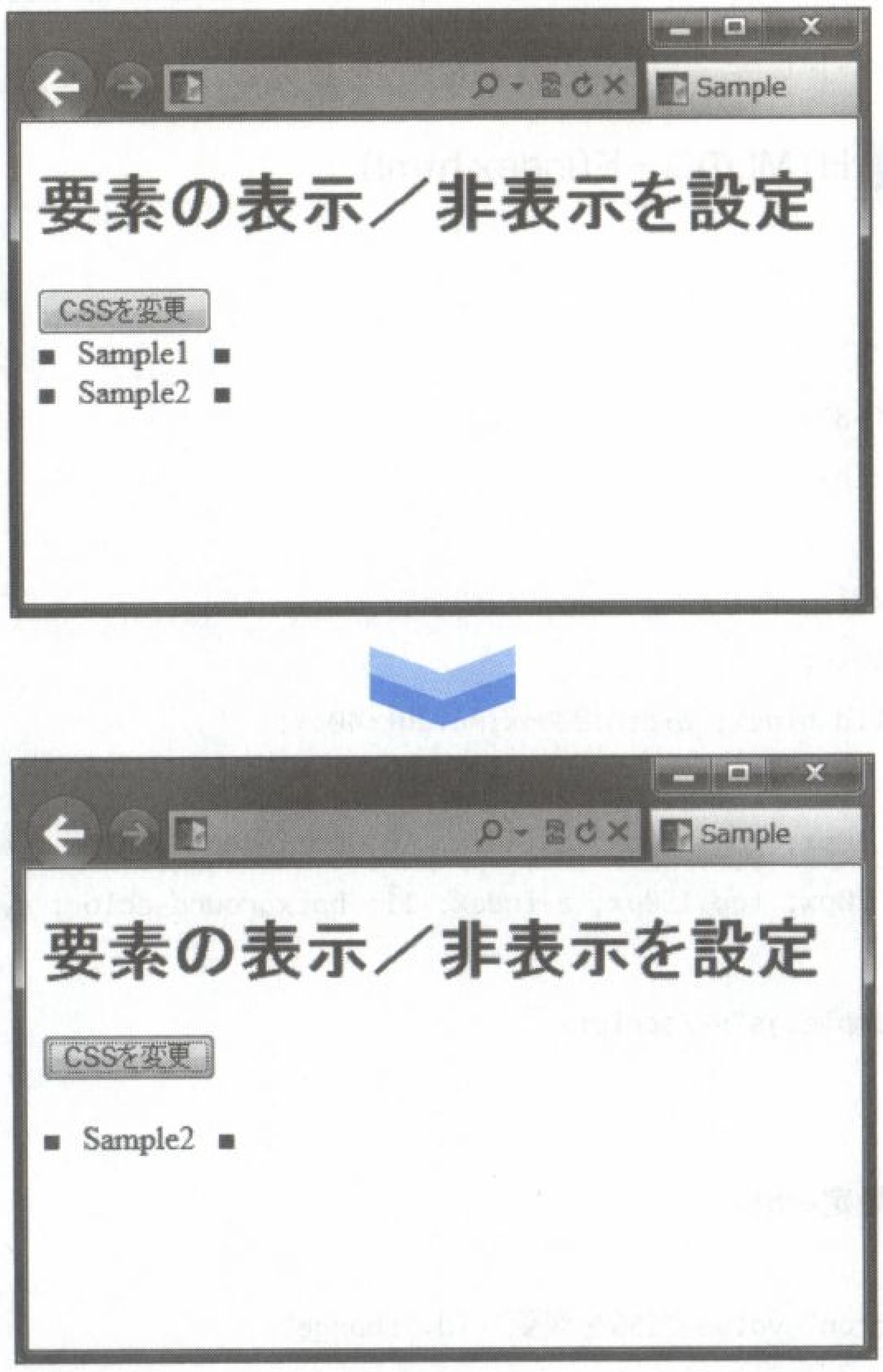
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の表示／非表示を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■ Sample1 ■</div>
    <div>■ Sample2 ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の表示方法を設定
    divElement.style.visibility = "hidden";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素が表示されなくなります。なお、「display」プロパティに「none」を設定したときとは異なり、他の要素がずれるようなことはありません。



ONEPOINT 要素の表示方法を変更するには スタイルシートの「visibility」プロパティを使う

要素の表示方法を変更するには、スタイルシートの「visibility」プロパティに、次の表に示す値（文字列）を設定します。なお、「display」プロパティとは異なり、非表示にすることで要素の位置がずれたりすることはありません（レイアウトは変化しない）。

文字列	説明
visible	表示
hidden	非表示

関連項目 ▶▶▶

● 要素の表示形式を変更する ……………………………………… p.926

SECTION-401

要素のZ座標を変更する

ここでは、スタイルシートで要素のZ座標を変更する方法について解説します。

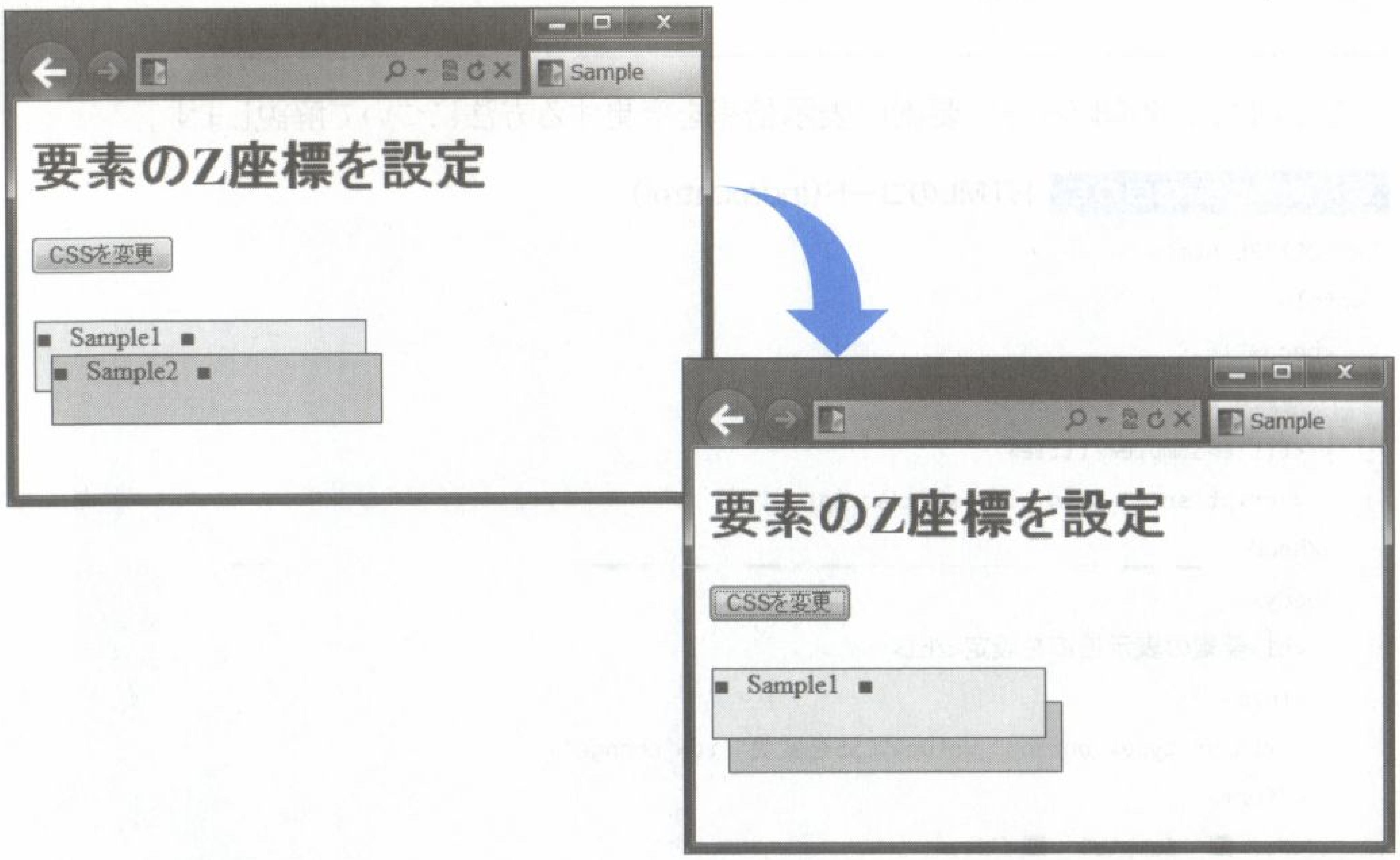
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      #block1, #block2 {
        position:absolute;
        border:1px solid black; width:200px;height:40px;
      }
      #block1 { left: 10px; top:130px; z-index: 10; background-color: yellow; }
      #block2 { left: 20px; top:150px; z-index: 11; background-color: cyan; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素のZ座標を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div id="block1">■  Sample1  ■</div>
    <div id="block2">■  Sample2  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素のZ座標を設定
    divElement.style.zIndex = 15;
  }, false);
}, false);
```

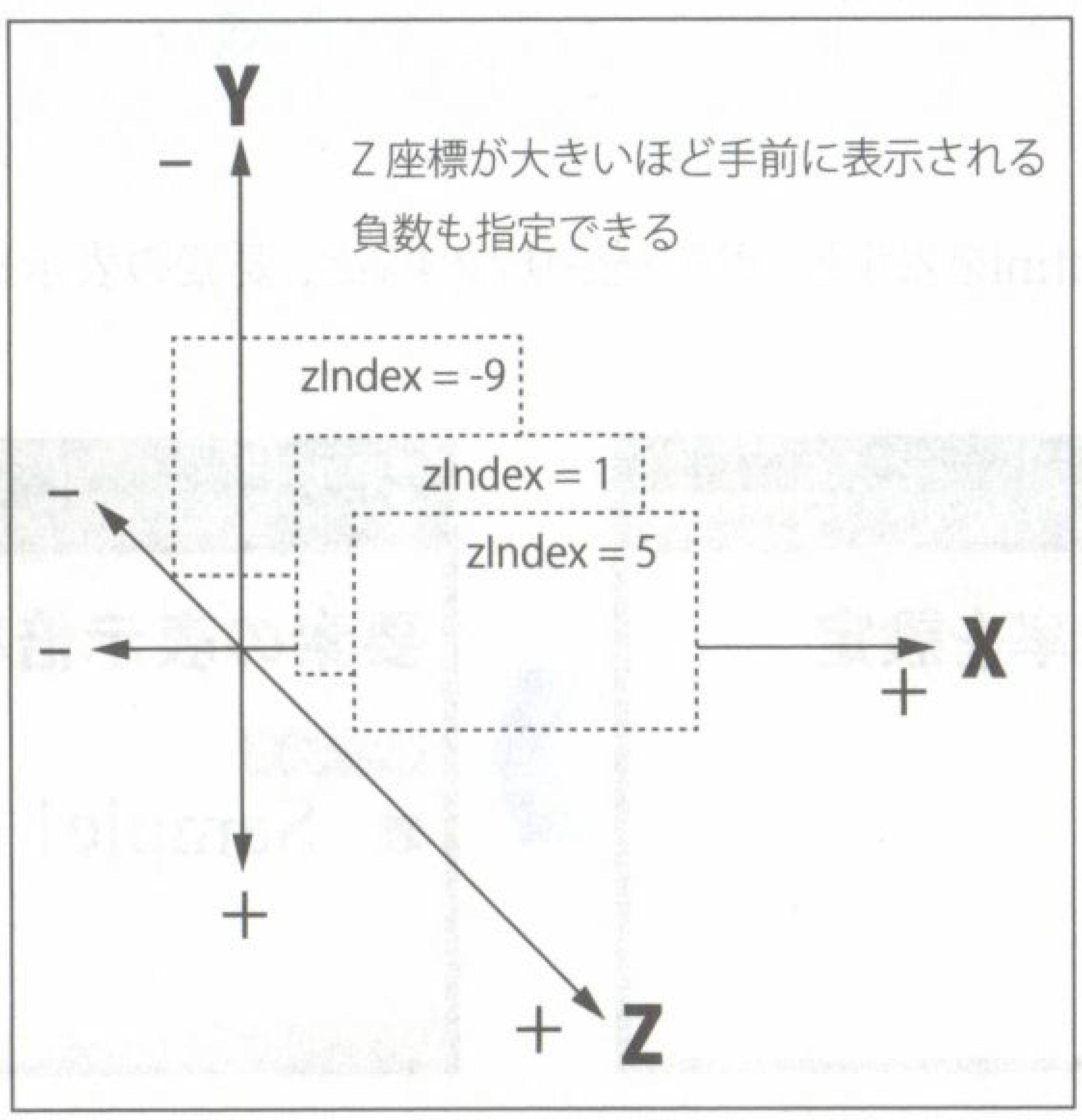
ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素のZ座標が変更され、奥にあった要素が手前に表示されます。



ONEPOINT **要素のZ座標を変更するには スタイルシートの「zIndex」プロパティを使う**

要素のZ座標を変更するには、スタイルシートの「zIndex」プロパティに、単位なしの数値を指定します。「zIndex」プロパティの値が大きいほど、手前に表示されます。負数の値も指定できますが、ブラウザによっては期待通り動作しない場合があります（次の図を参照）。

また、親要素の座標を継承する場合には、「inherit」の文字を指定します。なお、デフォルトでは、「auto」が設定されています。



SECTION-402

要素の表示倍率を変更する

ここでは、スタイルシートで要素の表示倍率を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の表示倍率を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■ Sample1 ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の表示倍率を設定
    divElement.style.zoom = 2.5;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の表示倍率が2.5倍に変更されます。



ONEPOINT 要素の表示倍率を変更するには スタイルシートの「zoom」プロパティを使う

要素の表示倍率を変更するには、スタイルシートの「zoom」プロパティに倍率を示す数値を設定します。値は「1.0」を指定すると100%サイズ、「0.5」なら50%サイズで表示されます。「2.0」であれば、200%サイズで表示されます。

また、「normal」の文字列を指定すると100%サイズになります。

COLUMN スマートフォンでの表示倍率

iPhone/Androidでは、次のように、「meta」要素に「initial-scale」を指定することで、最初の表示倍率を設定できます。なお、Windows Phoneでは、設定は反映されません。

```
<meta name="viewport" content="initial-scale=1.0">
```

また、ユーザーによる拡大縮小操作(スケーリング操作)を禁止するには、次のように指定します。

```
<meta name="viewport" content="user-scalable=no">
```

ユーザーによる拡大・縮小操作で最小の倍率を指定するには、次のように「minimum-scale」を使います。

```
<meta name="viewport" content="minimum-scale=1.0">
```

ユーザーによる拡大・縮小操作で最大の倍率を指定するには、次のようにminimum-scaleを使います。

```
<meta name="viewport" content="maximum-scale=2.0">
```

要素の角丸を変更する

ここでは、スタイルシートで要素の角丸を変更する方法について解説します。

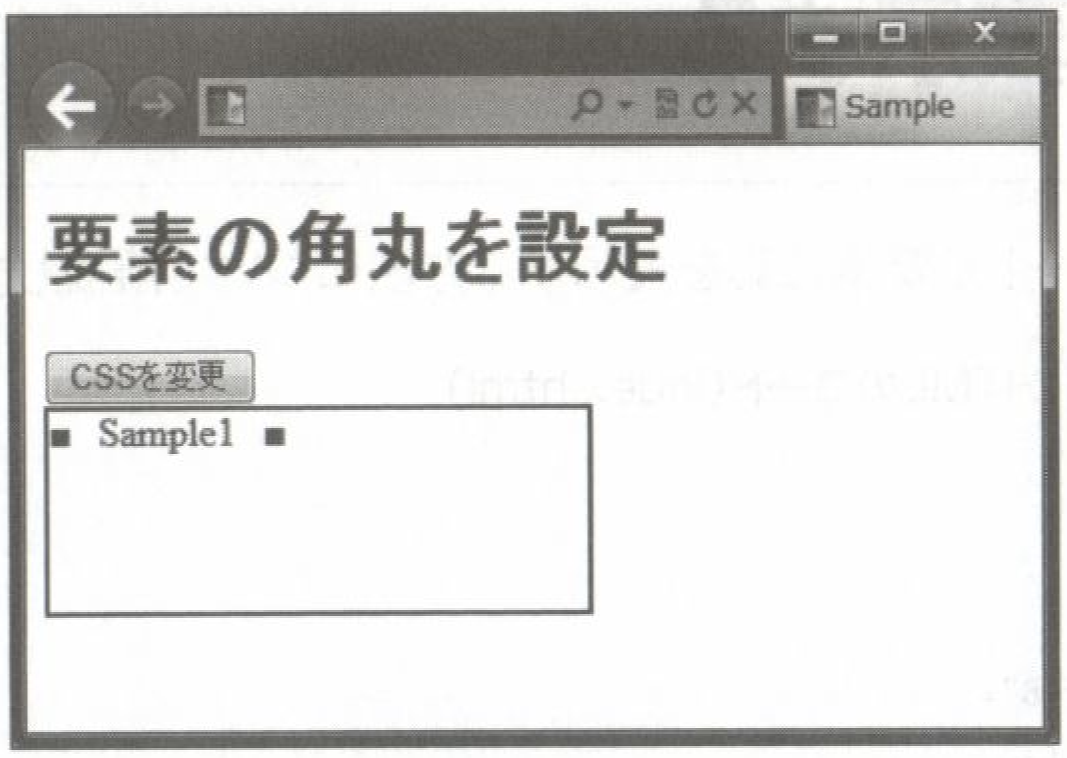
SAMPLE CODE　HTMLのコード(index.html)

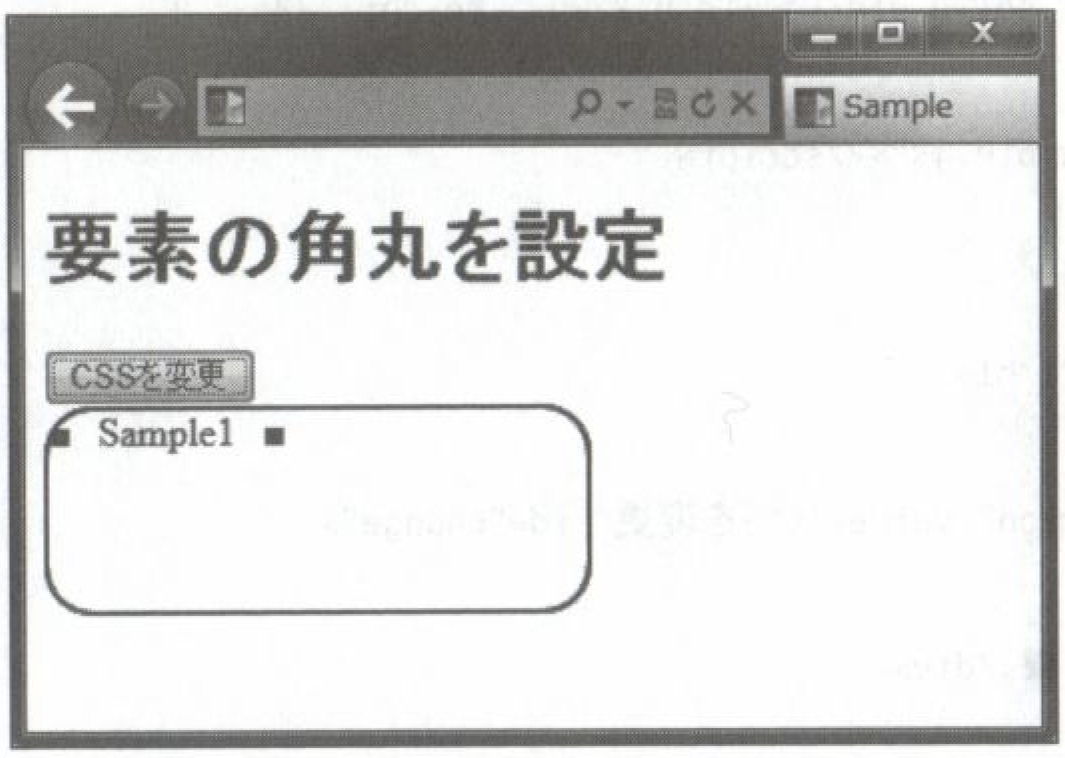
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { border:2px solid black; width:220px; height: 80px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の角丸を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■　Sample1　■</div>
  </body>
</html>
```

SAMPLE CODE　JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の角丸を設定
    divElement.style.borderRadius = "20px";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素の角が丸くなります。





ONEPOINT 要素の角丸を変更するには スタイルシートの「borderRadius」プロパティを使う

要素の角丸を変更するには、スタイルシートの「borderRadius」プロパティに角丸半径を示す単位付きの数値を設定します。設定できる値は、スタイルシートで設定できるものと同じになります。「"20px"」とすると、すべての角丸半径が20pxになります。「"20px 10px"」とすると、左上と右下が20pxに、右上と左下が10pxの角丸半径になります。

また、角丸を個別に指定する場合は、次の表に示すプロパティに単位付き数値を指定します。

プロパティ	説明
borderTopLeftRadius	左上の角
borderTopRightRadius	右上の角
borderBottomLeftRadius	左下の角
borderBottomRightRadius	右下の角

SECTION-404

要素に影を表示する

ここでは、スタイルシートで要素に影を表示する方法について解説します。

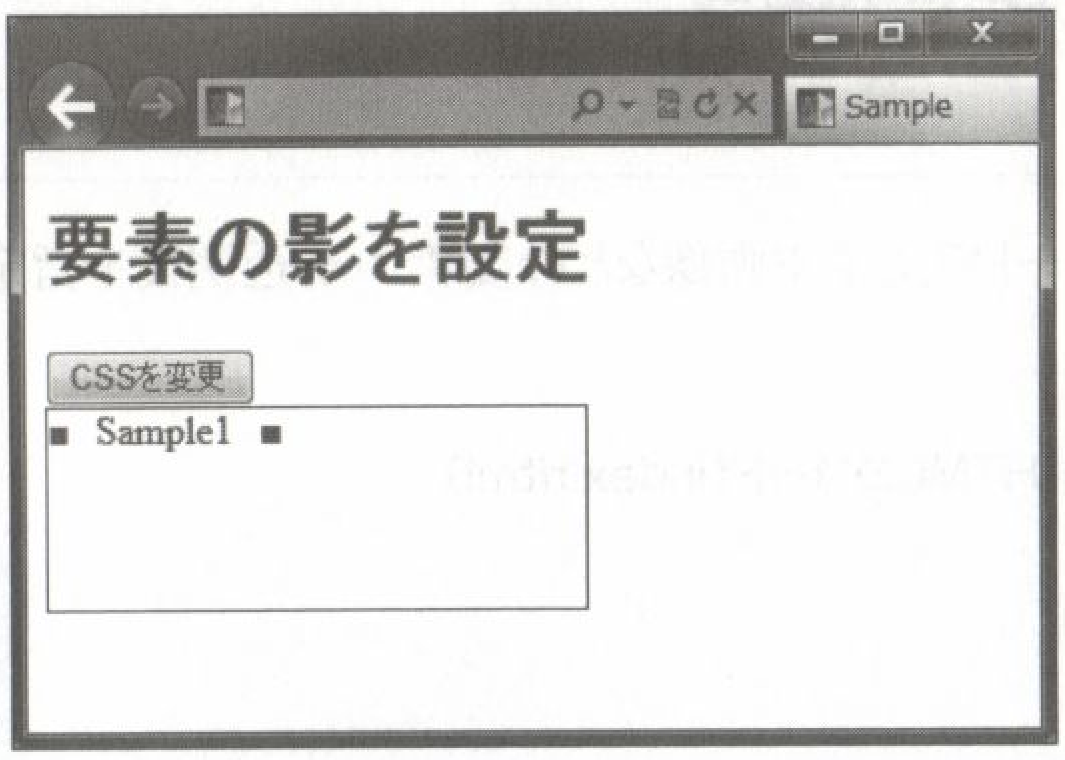
SAMPLE CODE HTMLのコード(index.html)

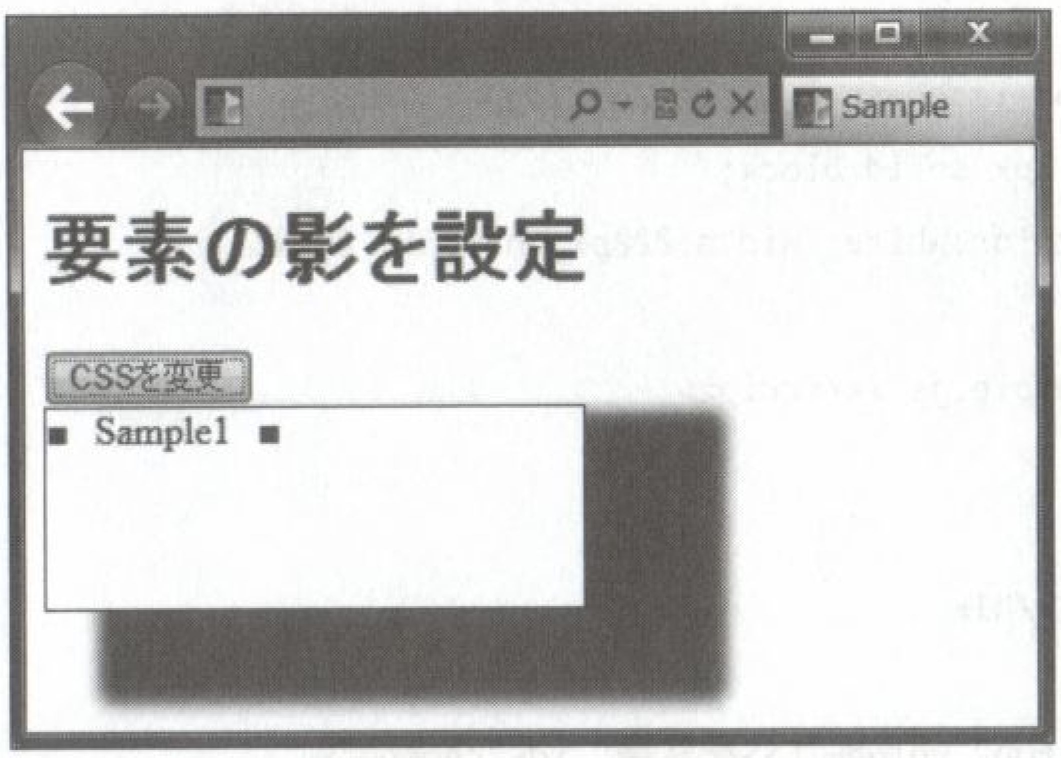
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { border:1px solid black; width:220px; height: 80px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素の影を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■  Sample1  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の影を設定
    divElement.style.boxShadow = "40px 20px 10px 15px blue";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素に青い影が表示されます。





ONEPOINT **要素に影を表示するには スタイルシートの「boxShadow」プロパティを使う**

要素に影を表示するには、スタイルシートの「boxShadow」プロパティに影を示す値を設定します。複数の値をまとめて指定します。その際、「影の横のずれ具合」「影の縦のずれ具合」「影のぼかし幅」「影の広がり幅」「色」の順番で影の位置や色などを指定し、「boxShadow」プロパティに文字列として一括で設定します。

不透明度を指定する

ここでは、スタイルシートで文字や画像などの要素の不透明度を指定する方法について解説します。

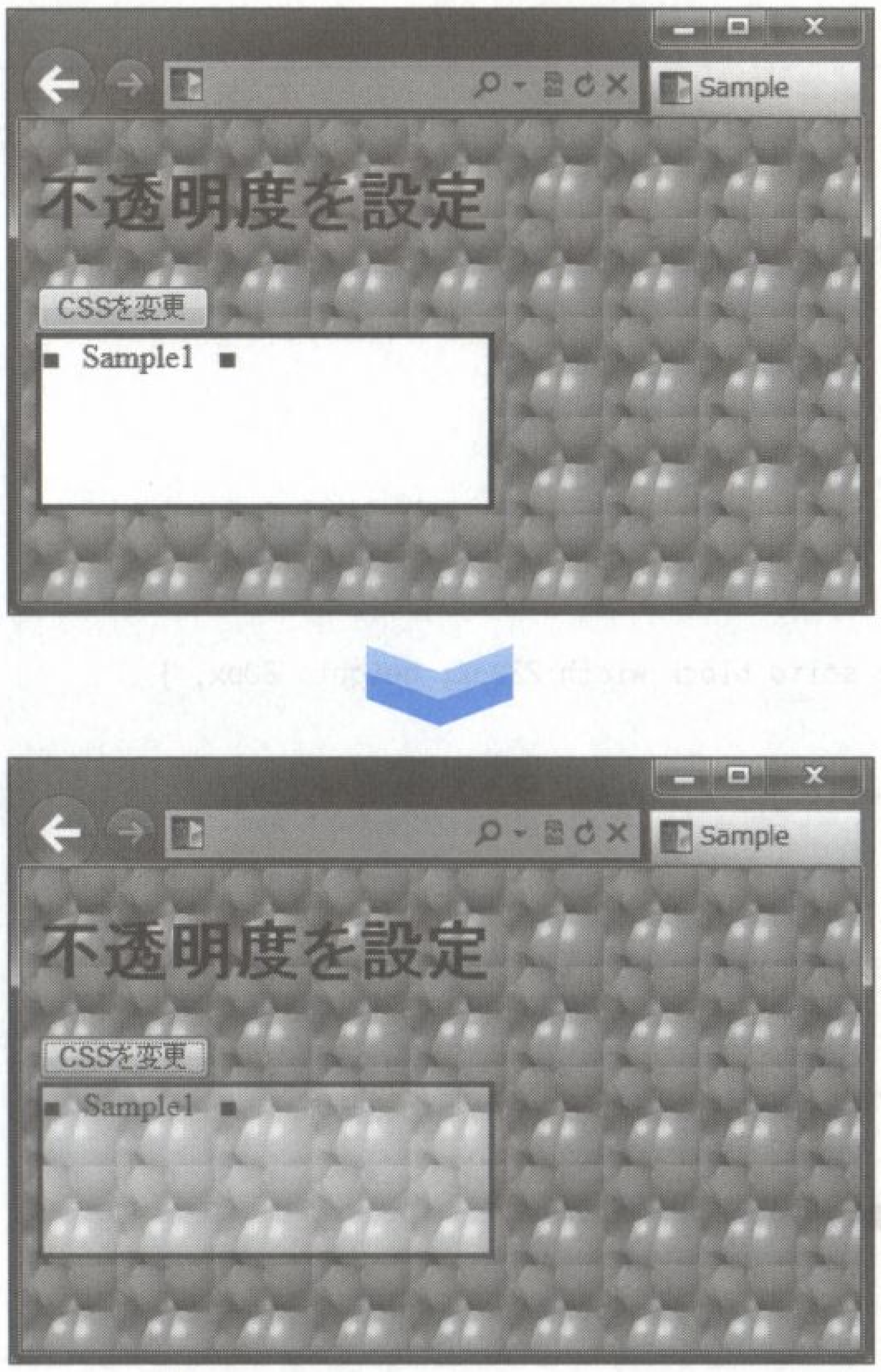
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      body { background-image: url(images/bg.jpg) }
      div {    border:3px solid black;
          background-color:white; width:220px; height: 80px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>不透明度を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■  Sample1  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素の不透明度を設定
    divElement.style.opacity = 0.5;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、ボックス部分の不透明度が50%になります。



ONEPOINT　不透明度を指定するにはスタイルシートの「opacity」プロパティを使う

文字や画像などの不透明度を指定するには、スタイルシートの「opacity」プロパティに、「0.0」～「1.0」までの数値を指定します。数値は大きいほど、不透明になります。つまり、「0」なら完全な透明になって要素は見えなくなり、「1.0」なら完全に見える状態になります。

また、「opacity」プロパティを操作せずに不透明度を指定するは、「color」プロパティや「backgroundColor」プロパティに、rgba形式かhsla形式で指定します（887ページ参照）。

SECTION-406

リサイズ処理を指定する

ここでは、スタイルシートでリサイズ処理を指定する方法について解説します。

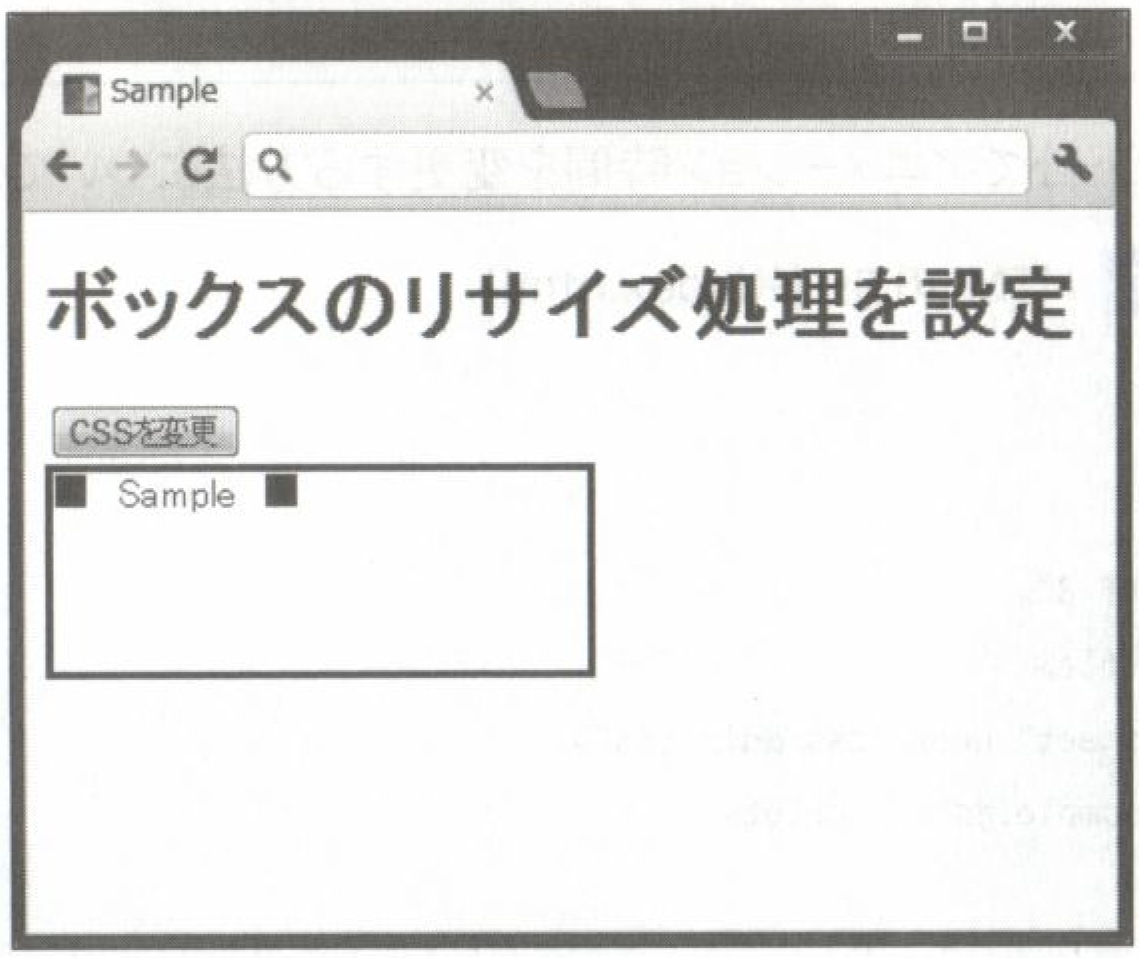
SAMPLE CODE HTMLのコード(index.html)

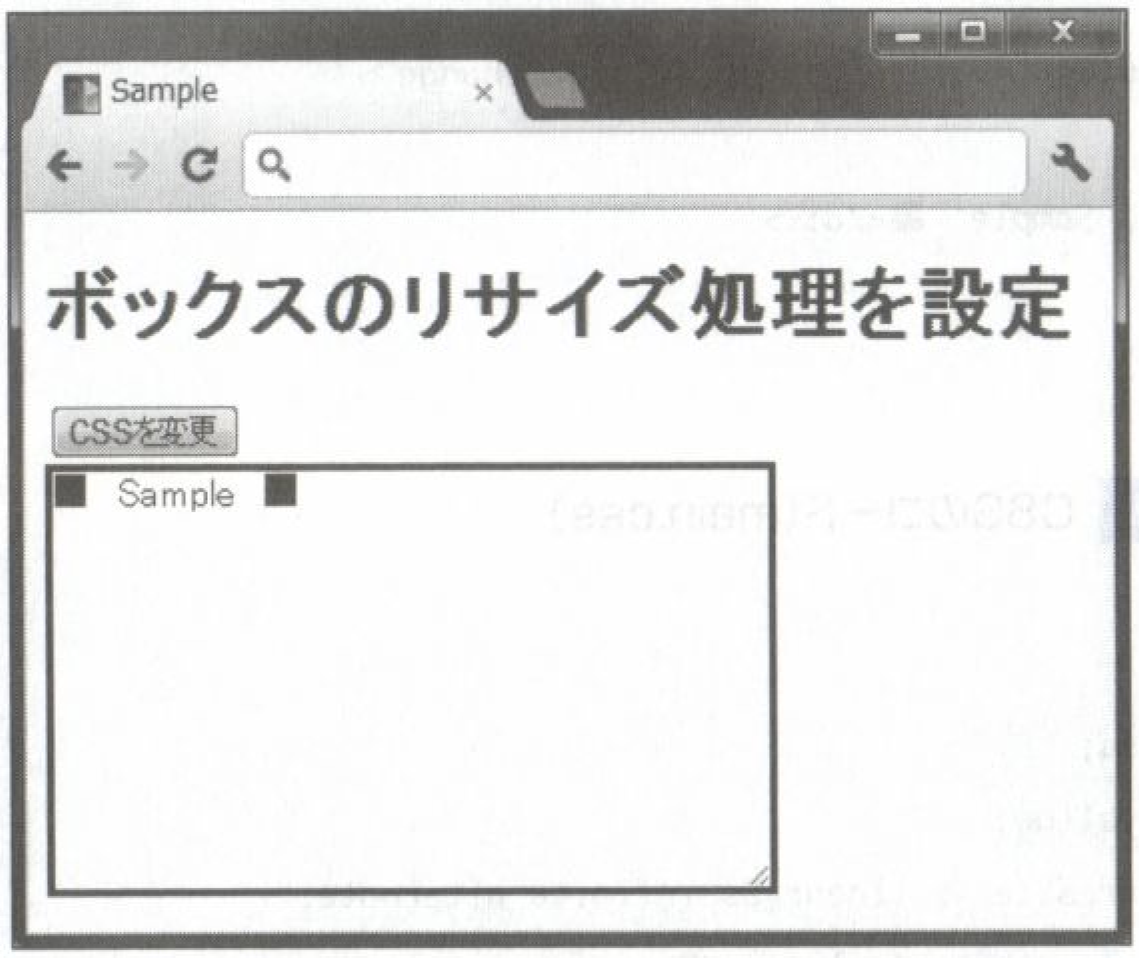
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <style>
      div { border:3px solid black;width:220px; height: 80px; }
    </style>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>ボックスのリサイズ処理を設定</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div>■  Sample  ■</div>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // div要素を読み出し
    var divElement = document.getElementsByTagName("div")[0];
    // 要素のリサイズ処理を設定
    divElement.style.resize = "both";
    divElement.style.overflow = "auto";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、ボックス部分の右下にリサイズボタンが追加されリサイズ処理が可能になります。





ONEPOINT 要素をリサイズするにはスタイルシートの「resize」プロパティを使う

要素をリサイズするには、スタイルシートの「resize」プロパティに、次の表に示す値（文字列）を設定します。また、同時に「overflow」プロパティに「"auto"」を設定する必要があります。

値	説明
none	リサイズしない
both	縦横ともリサイズ可能にする
horizontal	横方向のみリサイズ可能にする
vertical	縦方向のみリサイズ可能にする

SECTION-407

アニメーション時間を変更する

ここでは、スタイルシートでアニメーション時間を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素をアニメーション</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div id="box">■ Sample ■</div>
  </body>
</html>
```

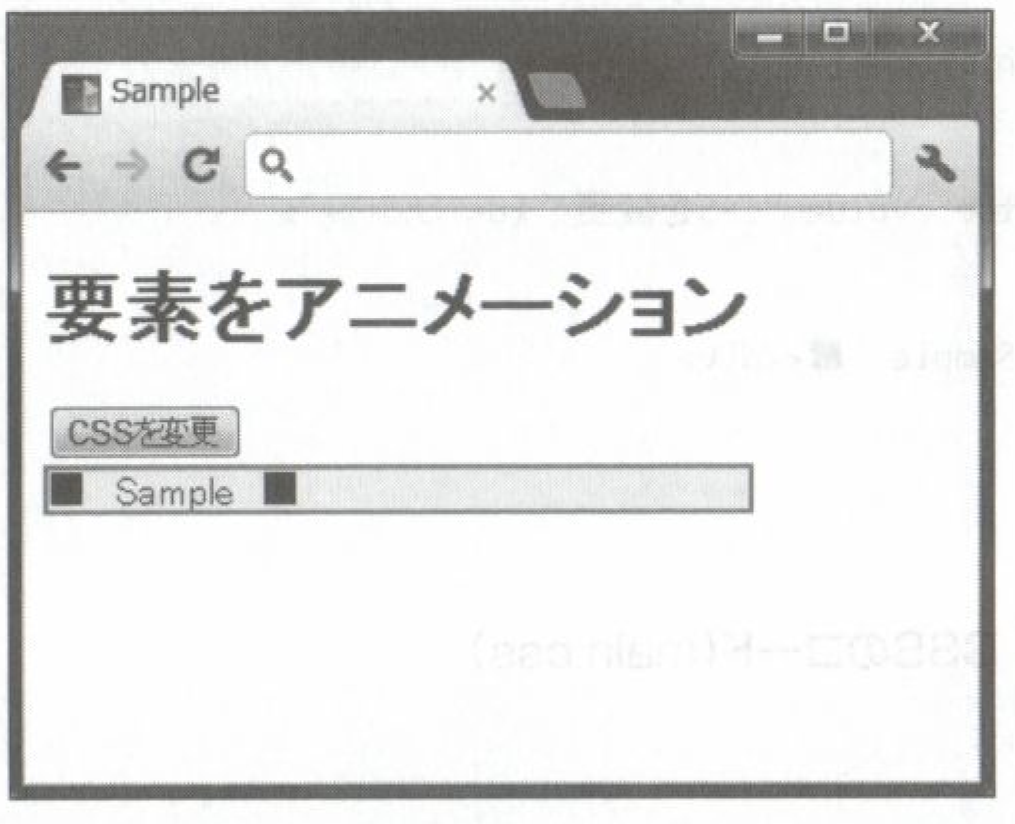
SAMPLE CODE CSSのコード(main.css)

```
#box {
  width : 300px;
  border:2px solid red;
  background-color: yellow;
  -moz-animation: boxresize 4s linear 0s infinite alternate;
  -webkit-animation: boxresize 4s linear 0s infinite alternate;
  animation: boxresize 4s linear 0s infinite alternate;
}
@keyframes boxresize {
  0% { width: 100px; }
  100% { width:320px; }
}
@-moz-keyframes boxresize {
  0% { width: 100px; }
  100% { width:320px; }
}
@-webkit-keyframes boxresize {
  0% { width: 100px; }
  100% { width:320px; }
}
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // アニメーションしているdiv要素を取得
    var animeBox = document.getElementById("box");
    // アニメーションの速度を変更(0.5秒にする)
    animeBox.style.WebkitAnimationDuration = "500ms";
    animeBox.style.MozAnimationDuration = "500ms";
    animeBox.style.animationDuration = "500ms";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ボタンをクリックすると、要素のアニメーション時間が変更されます。



ONEPOINT アニメーション時間を変更するには スタイルシートの「animationDuration」プロパティを使う

要素のアニメーションが行われる際の時間を変更するには、スタイルシートの「animation Duration」プロパティに、時間を示す文字列を指定します。たとえば、「"2s"」であれば、2秒でアニメーションが行われます。「"500ms"」であれば、500ミリ秒(0.5秒)でアニメーションが行われます。

なお、プロパティ名にベンダープレフィックスを付けないと、動作しない場合があります。Firefoxであれば「Moz」、SafariやGoogle Chromeであれば「webkit」(「Webkit」でも動作)、Operaの場合は「o」(オー)を付加します。をプロパティ名の前に付けます。サンプルでは、それぞれ、「MozAnimationDuration」「WebkitAnimationDuration」としています。

SECTION-408

3D変形を行う

ここでは、スタイルシートで表示されている要素に対し、3D変形を行う方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>要素を3D変形</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div id="box">■  Sample  ■</div>
  </body>
</html>
```

SAMPLE CODE CSSのコード(main.css)

```
#box {
  width : 320px;
  height : 240px;
  color: white;
  border:2px solid red;
  background-image: url(sample.jpg);
}
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // 3D変形するdiv要素を取得
    var animeBox = document.getElementById("box");
    // 3D変形
    var cssText = "matrix(0.5, -0.8, 0.2, 0.25, 50, 0) rotate(90deg)";
    animeBox.style.transform = cssText;
    animeBox.style.webkitTransform = cssText;
    animeBox.style.MozTransform = cssText;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、マトリクスによる変形を行った後に90度回転を行った結果が表示されます。要素内の文字や画像が同じように変形処理されます。



ONEPOINT　3D変形を行うにはスタイルシートの「transform」プロパティを使う

要素をスタイルシートを使って3D変形を行うには、スタイルシート「transform」プロパティに、次の表に示すいずれかのパラメータ(文字列)を設定します。また、半角空白で区切って列記することもできます。

パラメータ	説明
matrix(横の倍率, 縦の傾斜率, 横の傾斜率, 縦の倍率, 横の移動距量, 縦の移動量)	マトリクスによる変形
scale(横の倍率, 縦の倍率)	拡大・縮小(スケーリング)
rotate(回転角度)	回転
skew(横の傾斜率, 縦の傾斜率)	傾斜

なお、プロパティ名にベンダープレフィックスを付けないと、動作しない場合があります。Firefoxであれば「Moz」、SafariやGoogle Chromeであれば「webkit」(「Webkit」でも動作)をプロパティ名の前に付けます。サンプルでは、それぞれ、「MozTransform」「webkitTransform」としています。

SECTION-409

トランジションを設定する

ここでは、要素に対してトランジションを設定する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>トランジション</h1>
    <form>
      <input type="button" value="CSSを変更" id="change">
    </form>
    <div id="box">■ Sample ■</div>
  </body>
</html>
```

SAMPLE CODE CSSのコード(main.css)

```
#box {
  width : 320px;
  height : 240px;
  border:2px solid red;
  -webkit-transition: height 3s linear 1s;
  -moz-transition: height 3s linear 1s;
  transition: height 3s linear 1s;
}
#box:hover {
  height: 40px;
}
```

SAMPLE CODE JavaScriptのコード(sample.js)

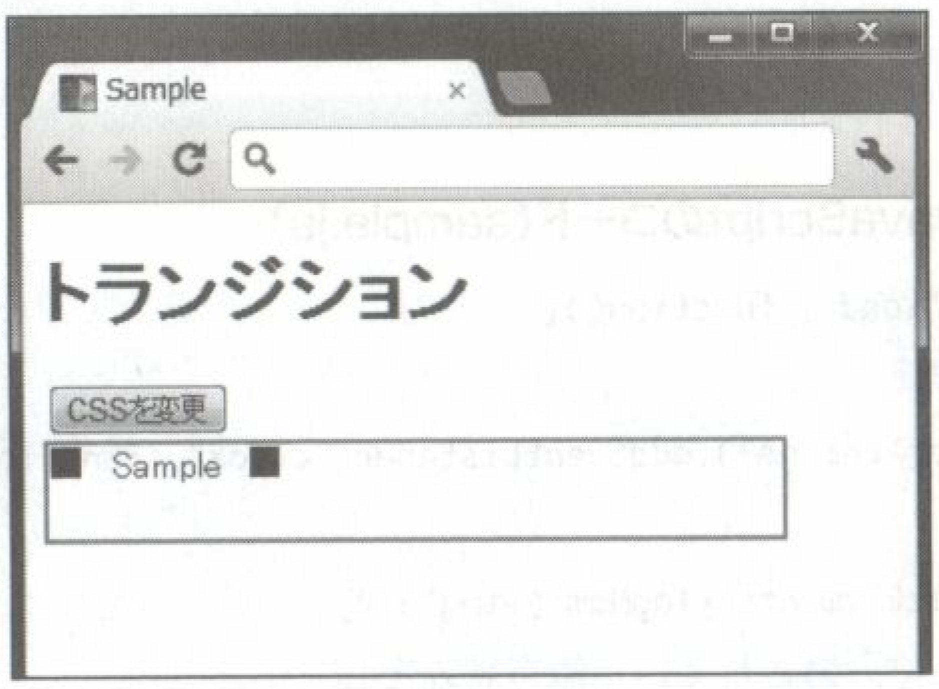
```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // トランジション(切り替え)を行うdiv要素を取得
    var animeBox = document.getElementById("box");
    // トランジションを設定
    var cssText = "height 0.25s linear";
    animeBox.style.transition = cssText;
```

```
    animeBox.style.webkitTransition = cssText;
    animeBox.style.MozTransition = cssText;
  }, false);
}, false);
```

ブラウザでindex.htmlを表示すると、赤枠の要素にはトランジションが設定されており、マウスを乗せると1秒後にトランジションが開始されます。



ボタンをクリックするとトランジションの速度が変更され、マウスを乗せると0.25秒後にトランジションが開始されます。



ONEPOINT **トランジションを設定するには スタイルシートの「transition」プロパティを使う**

要素にトランジションを設定するには、スタイルシートの「transition」プロパティに、変更するプロパティや速度を一括して文字列で指定します。複数のプロパティを変更する場合は、「"height 0.25s linear, width 1s linear"」のように、「,」(カンマ)で区切って記述します。

トランジションを変更する場合は、直接、「transition」プロパティに設定するよりも、CSSでクラスを定義して切り換える方が簡単です。

なお、プロパティ名は、ベンダープレフィックスを付けないと動作しない場合があります。Firefoxであれば「Moz」、SafariやGoogle Chromeであれば「webkit」(Webkitでも動作)をプロパティ名の前に付けます。サンプルでは、それぞれ、「MozTransition」「webkitTransition」としています。

SECTION-410

要素にグレーフィルタを適用する

ここでは、スタイルシートで要素にグレーフィルタを適用する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素にグレーフィルタを適用</h3>
    <form>
      値：<input type="text" value="100%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // グレーフィルタを設定
    img.style.webkitFilter = "grayscale("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、グレースケールの割合(「0%」~「100%」、または、「0.0」~「1.0」の数値)を入力してボタンをクリックすると、画像がグレースケールになります。



ONEPOINT

グレースケールにするにはスタイルシートの「filter」プロパティに「"grayscale(値)"」の文字列を指定する

要素をグレースケールにするには、スタイルシートの「filter」プロパティに「"grayscale(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は0%～100%までの範囲、もしくは0.0～1.0の範囲で指定します。値が大きくなるほどフィルタ効果の度合いが強くなります。

COLUMN

「filter」プロパティのプロパティ名について

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- グレースケール画像にする …… p.480
- 要素をセピア調にする …… p.956
- 要素にトーン(色相変換)フィルタを適用する …… p.958
- 要素に色反転フィルタを適用する …… p.960
- 要素の輝度(明るさ)を変更する …… p.962
- 要素のコントラストを変更する …… p.965
- 要素の彩度を変更する …… p.968
- 要素にぼかしフィルタを適用する …… p.971

SECTION-411

要素をセピア調にする

ここでは、スタイルシートで要素をセピア調にする方法について解説します。

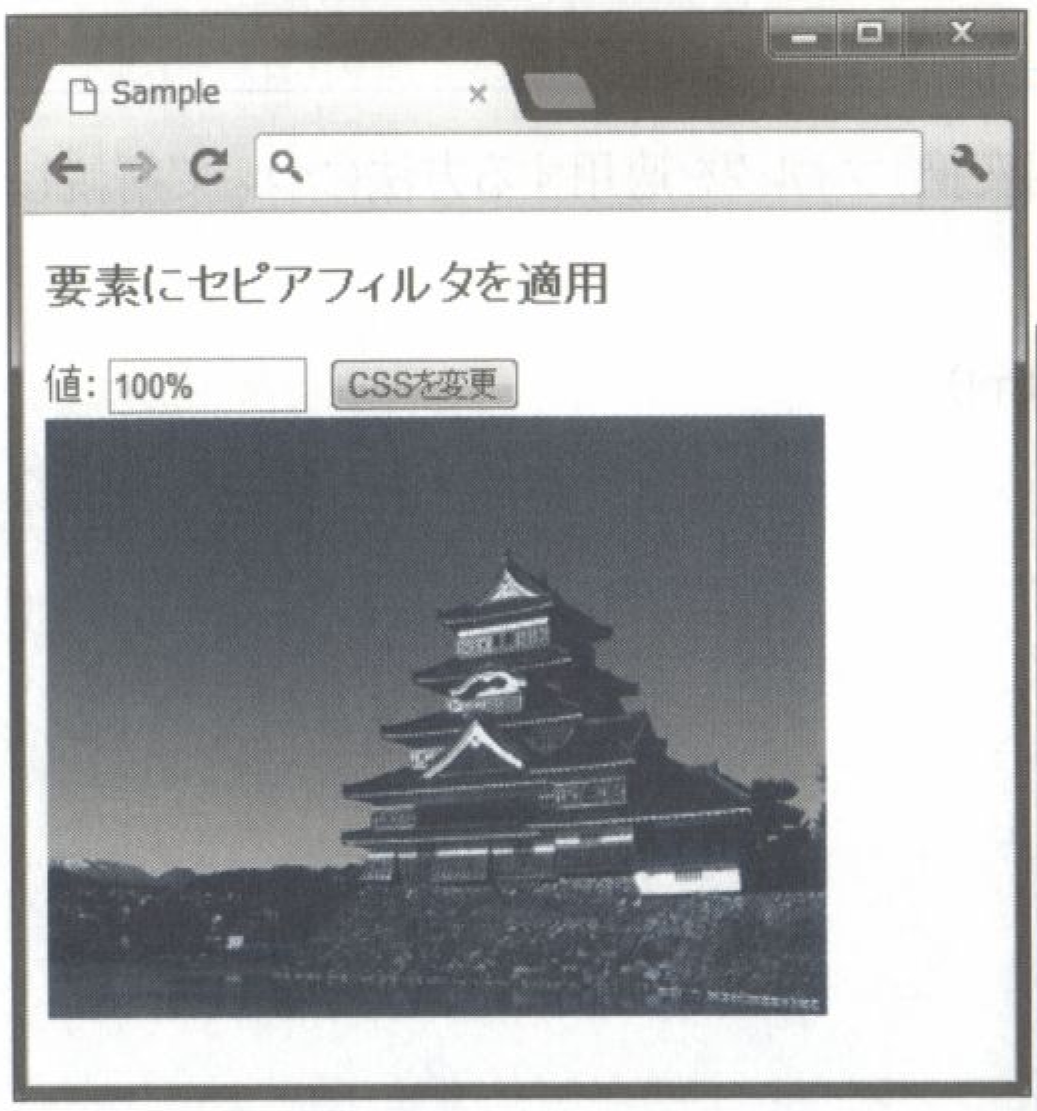
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素にセピアフィルタを適用</h3>
    <form>
      値 :<input type="text" value="100%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // セピアフィルタを設定
    img.style.webkitFilter = "sepia("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し。セピアの適用割合(「0%」~「100%」、または、「0.0」~「1.0」の数値)を入力してボタンをクリックすると、画像がセピア調になります。





ONEPOINT セピア調にするにはスタイルシートの「filter」プロパティに「"sepia(値)"」の文字列を指定する

要素をセピア調にするには、スタイルシートの「filter」プロパティに「"sepia(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」~「100%」、もしくは、「0.0」~「1.0」の範囲で指定します。値が大きくなるほど、フィルタ効果の度合いが強くなります。

COLUMN 「filter」プロパティのプロパティ名について

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- 要素にグレーフィルタを適用する ………… p.954
- 要素にトーン(色相変換)フィルタを適用する ………… p.958
- 要素に色反転フィルタを適用する ………… p.960
- 要素の輝度(明るさ)を変更する ………… p.962
- 要素のコントラストを変更する ………… p.965
- 要素の彩度を変更する ………… p.968
- 要素にぼかしフィルタを適用する ………… p.971

SECTION-412

要素にトーン(色相変換)フィルタを適用する

ここでは、スタイルシートで要素にトーン(色相変換)フィルタを適用する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素にトーンフィルタを適用</h3>
    <form>
      値 :<input type="text" value="120" id="filterValue" size="8">deg(0～360)
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // トーン(色相変換)フィルタを設定
    img.style.webkitFilter = "hue-rotate("+val+"deg)";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、トーン(色相変換)フィルタの割合(「0%」～「100%」、または、「0.0」～「1.0」の数値)を入力してボタンをクリックすると、画像のトーンが変化します。





ONEPOINT トーン(色相)を変更するにはスタイルシートの「filter」プロパティに「"hue-rotate(値)"」の文字列を指定する

要素のトーン(色相)を変更にするには、スタイルシートの「filter」プロパティに「"hue-rotate(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」~「100%」、もしくは、「0.0」~「1.0」の範囲で指定します。値が大きくなるほど、フィルタ効果の度合いが強くなります。

COLUMN 「filter」プロパティのプロパティ名について

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- 画像を特定の色相(トーン)にする …… p.499
- 要素にグレーフィルタを適用する …… p.954
- 要素をセピア調にする …… p.956
- 要素に色反転フィルタを適用する …… p.960
- 要素の輝度(明るさ)を変更する …… p.962
- 要素のコントラストを変更する …… p.965
- 要素の彩度を変更する …… p.968
- 要素にぼかしフィルタを適用する …… p.971

SECTION-413

要素に色反転フィルタを適用する

ここでは、スタイルシートで要素に色反転フィルタを適用する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素に色反転フィルタを適用</h3>
    <form>
      値：<input type="text" value="100%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // 色反転フィルタを設定
    img.style.webkitFilter = "invert("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィルタ効果の割合(0%~100%または0.0~1.0の数値)を入力してボタンをクリックすると、画像の色が反転します。



ONEPOINT　**色は反転させるにはスタイルシートの「filter」プロパティに「"invert(値)"」の文字列を指定する**

要素の色を反転させるにはスタイルシートの「filter」プロパティに「"invert(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」~「100%」、もしくは、「0.0」~「1.0」の範囲で指定します。値が大きくなるほどフィルタ効果の度合いが強くなります。

COLUMN　**「filter」プロパティのプロパティ名について**

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

SECTION-414

要素の輝度(明るさ)を変更する

ここでは、スタイルシートで要素の輝度(明るさ)を変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素に輝度調整フィルタを適用</h3>
    <form>
      値 :<input type="text" value="40%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

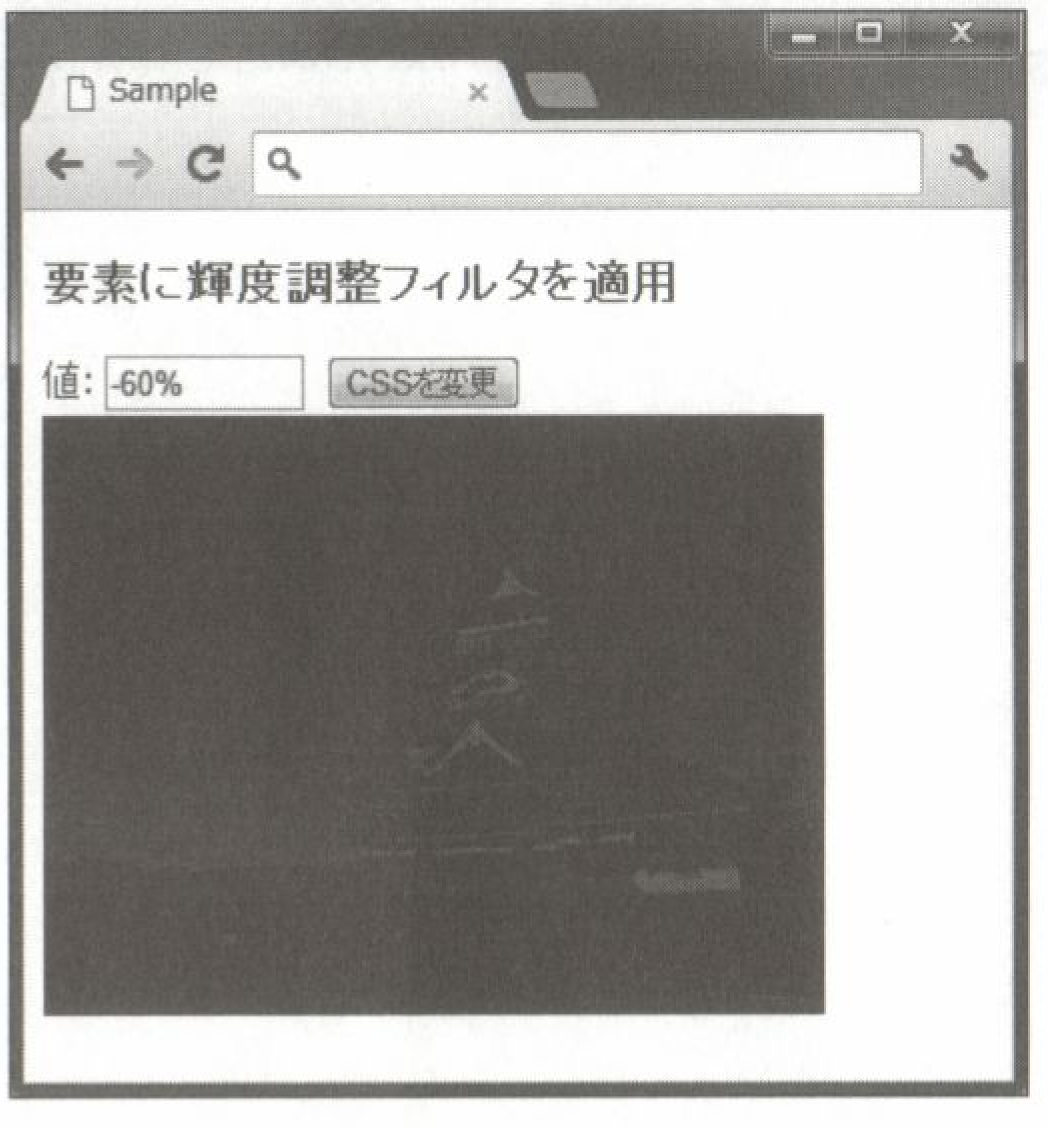
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // 輝度調整フィルタを設定
    img.style.webkitFilter = "brightness("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、輝度の強さ（「-100%」～「100%」、または、「0.0」～「1.0」の数値）を入力してボタンをクリックすると、画像の明るさが変化します。



負数の場合は画像が暗くなります。



ONEPOINT **明るさを調整するにはスタイルシートの「filter」プロパティに「"brightness（値）"」の文字列を指定する**

要素の明るさを調整するにはスタイルシートの「filter」プロパティに「"brightness（値）"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「-100%」～「100%」、もしくは、「-1.0」～「1.0」の範囲で指定します。値が大きくなるほど明るくなり、負数になると暗くなります。

COLUMN **「filter」プロパティのプロパティ名について**

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- 画像のガンマ補正を行う …… p.492
- 要素にグレーフィルタを適用する …… p.954
- 要素をセピア調にする …… p.956
- 要素にトーン（色相変換）フィルタを適用する …… p.958
- 要素に色反転フィルタを適用する …… p.960
- 要素のコントラストを変更する …… p.965
- 要素の彩度を変更する …… p.968
- 要素にぼかしフィルタを適用する …… p.971

SECTION-415

要素のコントラストを変更する

ここでは、スタイルシートで要素のコントラストを変更する方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素にコントラストフィルタを適用</h3>
    <form>
      値：<input type="text" value="150%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

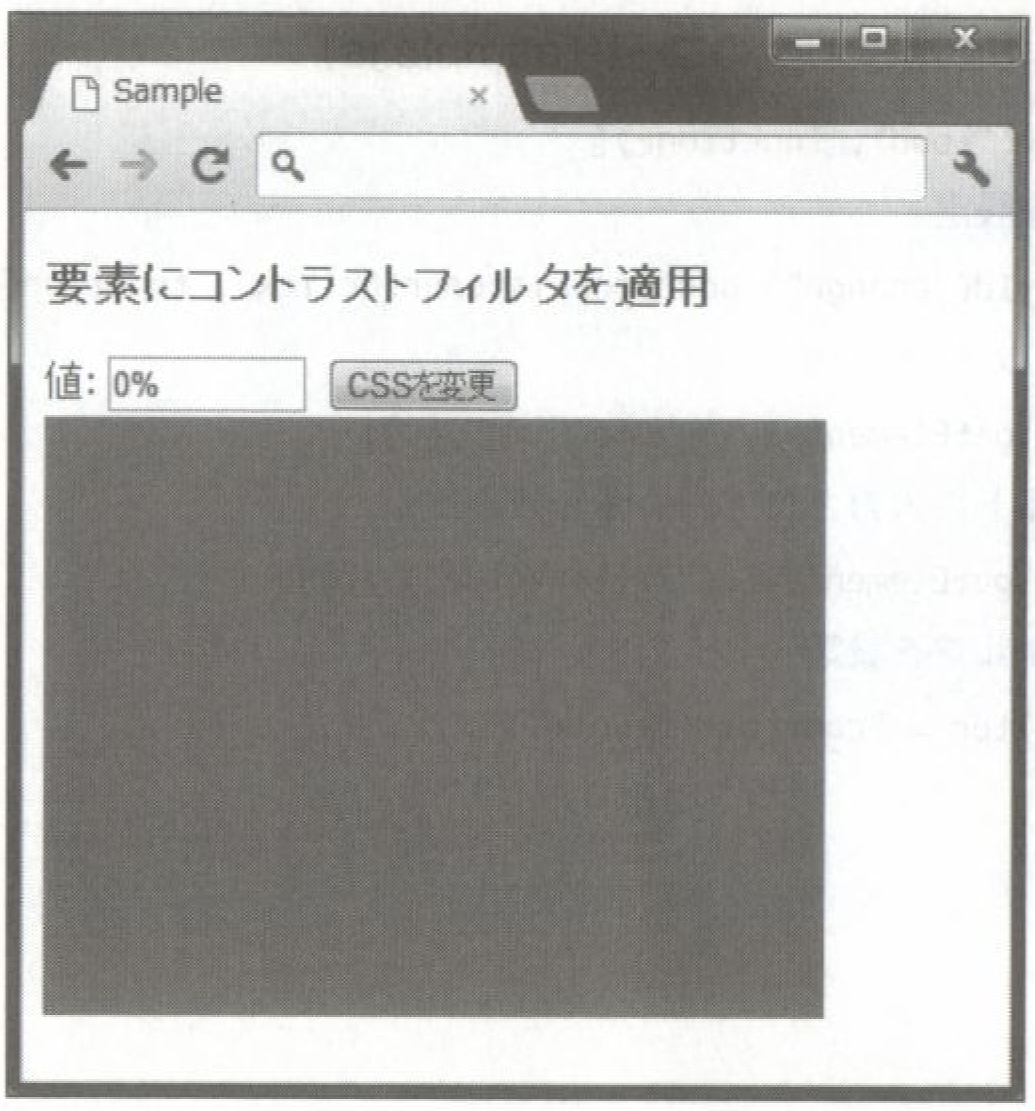
SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // コントラストフィルタを設定
    img.style.webkitFilter = "contrast("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィルタ効果の割合を入力してボタンをクリックすると、値が大きいほど、コントラストが強くなります。100%より大きな値を指定することもできます。



「0%」を設定すると、コントラストがなくなるため、画像は灰色になります。



ONEPOINT **コントラストを調整するにはスタイルシートの「filter」プロパティに「"contrast(値)"」の文字列を指定する**

要素のコントラストを調整するにはスタイルシートの「filter」プロパティに「"contrast(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」以上、もしくは、「0」以上の数値で指定します。値が大きくなるほど、コントラストが強くなります。「0%」を指定すると、まったくコントラストがなくなるため、要素の内容は灰色になります。

COLUMN **「filter」プロパティのプロパティ名について**

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- 要素にグレーフィルタを適用する ………… p.954
- 要素をセピア調にする ………… p.956
- 要素にトーン(色相変換)フィルタを適用する ………… p.958
- 要素に色反転フィルタを適用する ………… p.960
- 要素の輝度(明るさ)を変更する ………… p.962
- 要素の彩度を変更する ………… p.968
- 要素にぼかしフィルタを適用する ………… p.971

SECTION-416

要素の彩度を変更する

ここでは、スタイルシートで要素の彩度を変更する方法について解説します。

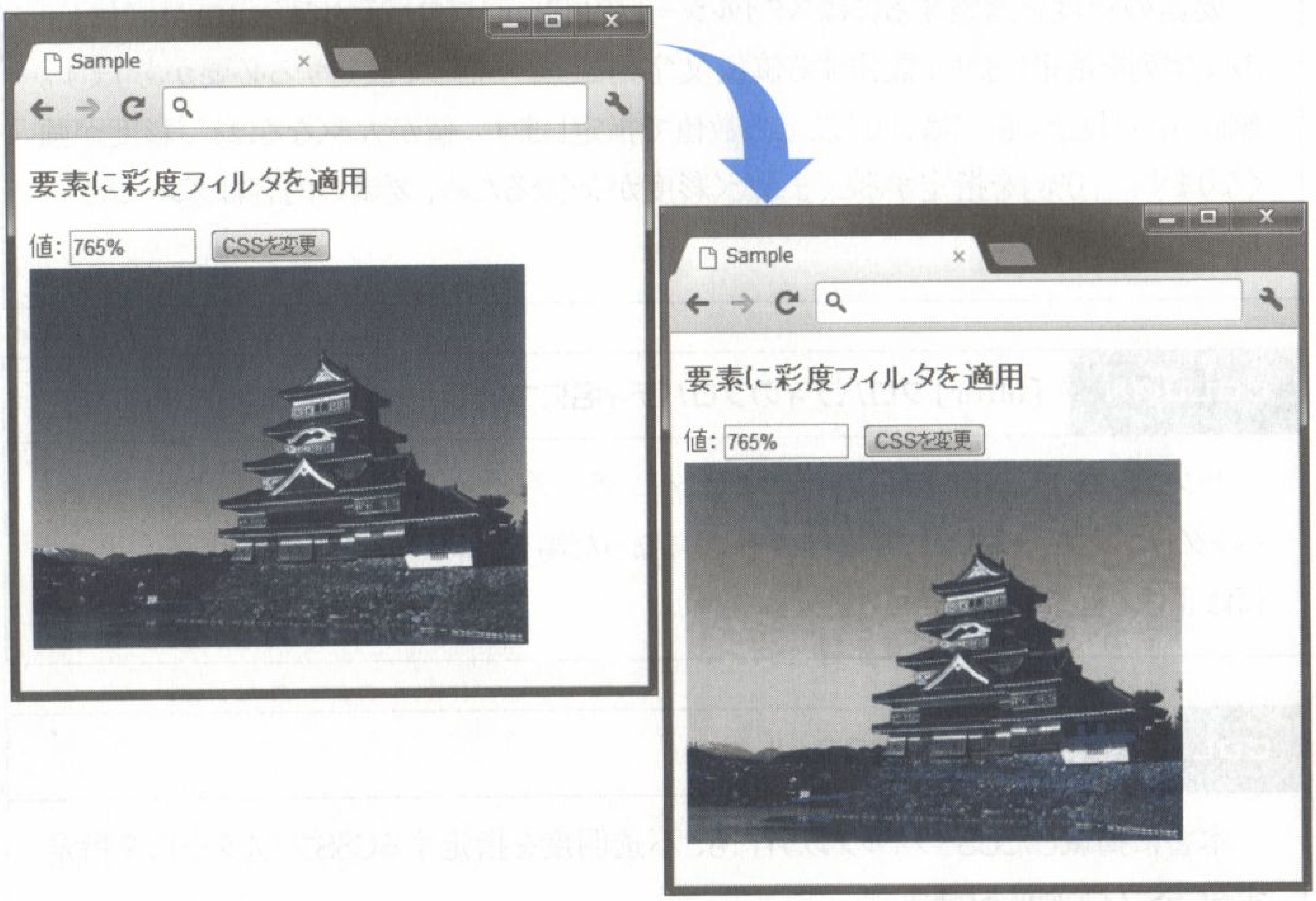
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素に彩度フィルタを適用</h3>
    <form>
      値：<input type="text" value="150%" id="filterValue" size="8">
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // 彩度フィルタを設定
    img.style.webkitFilter = "saturate("+val+")";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、フィルタ効果の割合を入力してボタンをクリックすると、値が大きいほど彩度が強くなります。100%より大きな値を指定することもできます。



「0%」を設定すると、彩度がなくなるため、画像はグレースケールになります。



ONEPOINT 彩度を調整するにはスタイルシートの「filter」プロパティに「"saturate(値)"」の文字列を指定する

要素の彩度を調整するにはスタイルシートの「filter」プロパティに「"saturate(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」以上、もしくは、「0」以上の数値で指定します。値が大きくなるほど、彩度が強くなります。「0%」を指定すると、まったく彩度がなくなるため、要素の内容はグレースケールと同じになります。

COLUMN 「filter」プロパティのプロパティ名について

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

COLUMN その他のCSSフィルタ

本書に掲載したCSSフィルタ以外にも、不透明度を指定するCSSフィルタや影を指定するCSSフィルタがあります。

要素の不透明度を指定するには、「filter」プロパティに「"opacity(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。値は「0%」~「100%」、もしくは、「0.0」~「1.0」の範囲で指定します。「0%」(または「0.0」)が完全に透明で、「100%」(または「1.0」)が完全に不透明になります。

要素に影を指定するには、「filter」プロパティに「"drop-shadow(横方向のオフセット 縦方向のオフセット ぼかしの半径 影の色)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。「横方向のオフセット」「縦方向のオフセット」「ぼかしの半径」はピクセル単位で指定します(pxを付けて指定する)。「影の色」には色を示すカラー名やカラーコードを指定します。指定可能なカラー形式については、887ページを参照してください。

なお、「-webkit-drop-shadow」のように、ベンダープレフィックスが必要な場合があります。

関連項目 ▶▶▶

SECTION-417

要素にぼかしフィルタを適用する

ここでは、スタイルシートで要素にぼかしフィルタを適用する方法について解説します。

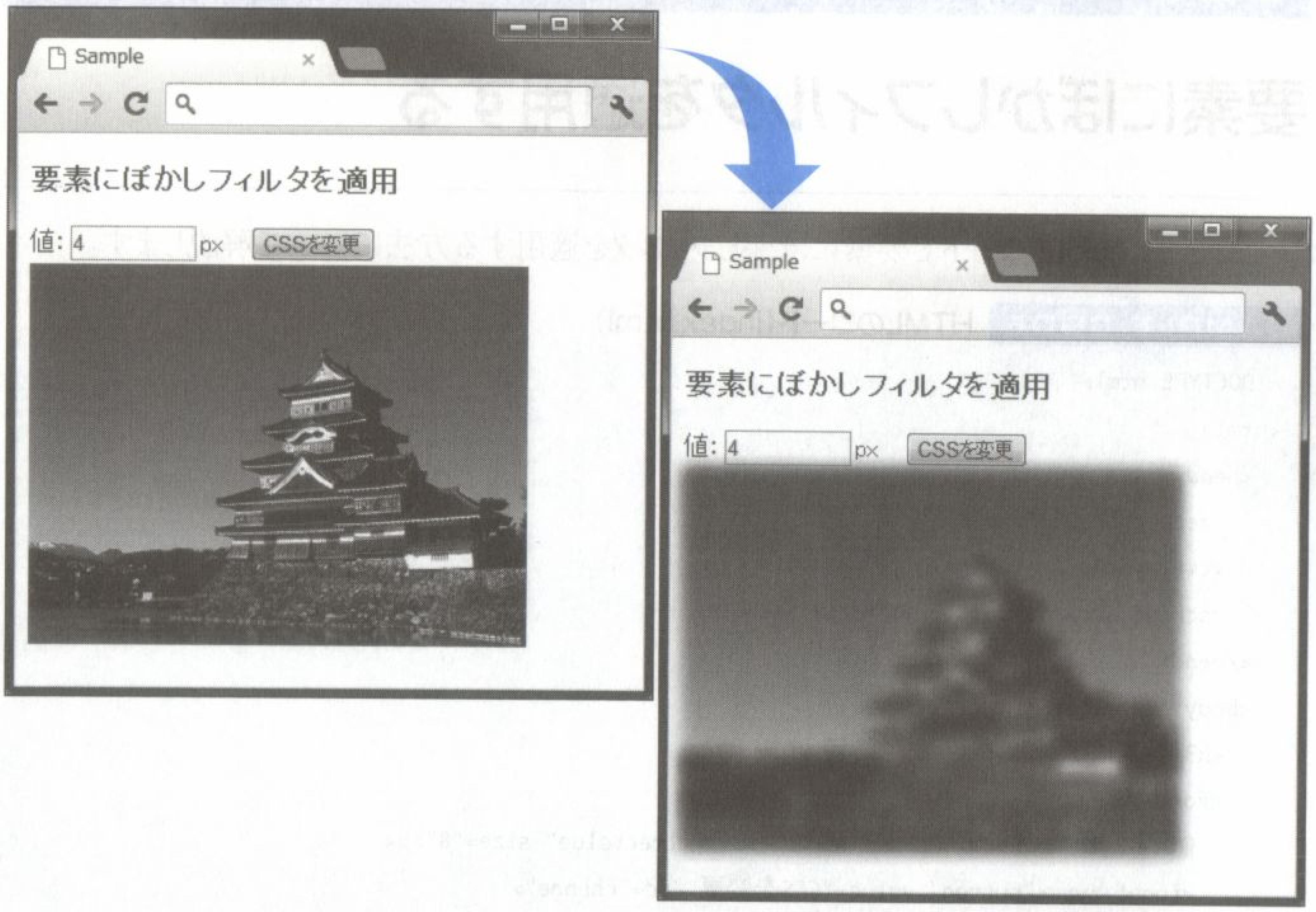
SAMPLE CODE HTMLのコード(index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h3>要素にぼかしフィルタを適用</h3>
    <form>
      値：<input type="text" value="2" id="filterValue" size="8">px
      <input type="button" value="CSSを変更" id="change">
    </form>
    <img src="images/sample.jpg" width="320" height="240">
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

```
window.addEventListener("load", function(){
  // ボタンにイベントを設定
  document.getElementById("change").addEventListener("click", function(){
    // img要素を読み出し
    var img = document.getElementsByTagName("img")[0];
    // テキストフィールドに入力されている値を読み出し
    var val = document.getElementById("filterValue").value;
    // ぼかしフィルタを設定
    img.style.webkitFilter = "blur("+val+"px)";
  }, false);
}, false);
```

ブラウザでindex.htmlを表示し、ぼかし幅(単位はピクセル)を入力してボタンをクリックすると、値が大きいほどぼかし具合が強くなります。



ONEPOINT ぼかすにはスタイルシートの「filter」プロパティに「"blur(値)"」の文字列を指定する

要素をぼかすにはスタイルシートの「filter」プロパティに「"blur(値)"」の文字列を指定します。設定する値も、文字列として一括して設定する必要があります。ぼかし幅をピクセルで指定します。その際、整数値だけでなく、「1.25px」といった小数値も指定することができます。なお、ぼかし効果は要素が表示されている範囲を超えて表示されるので、他の要素にもぼかした影響が広がることがあります。

COLUMN 「filter」プロパティのプロパティ名について

サンプルでは、「webkitFilter」のようにベンダープレフィックスが付いています。将来、ベンダープレフィックスなしで動作するようになった場合には、「webkitFilter」を「filter」に修正する必要があります。

関連項目 ▶▶▶

- 画像をぼかす …… p.488
- 要素にグレーフィルタを適用する …… p.954
- 要素をセピア調にする …… p.956
- 要素にトーン(色相変換)フィルタを適用する …… p.958
- 要素に色反転フィルタを適用する …… p.960
- 要素の輝度(明るさ)を変更する …… p.962
- 要素のコントラストを変更する …… p.965
- 要素の彩度を変更する …… p.968

SECTION-418

CSSのメディアクエリを調べる

ここでは、スタイルシートで利用されるメディアクエリを調べる方法について解説します。

SAMPLE CODE HTMLのコード(index.html)

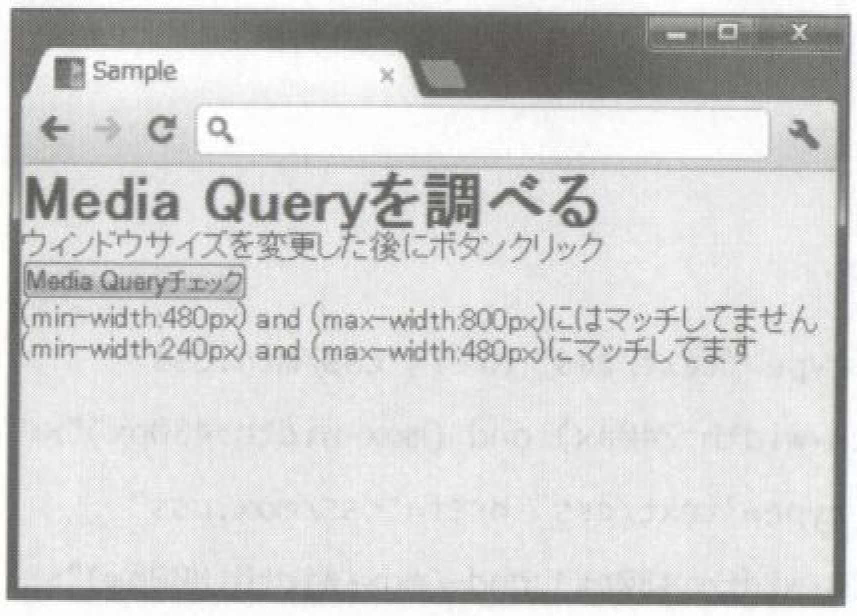
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sample</title>
    <link rel="stylesheet" type="text/css" href="css/min.css"
      media="screen and (min-width:240px) and (max-width:480px)">
    <link rel="stylesheet" type="text/css" href="css/max.css"
      media="screen and (min-width:480px) and (max-width:800px)">
    <script src="js/sample.js"></script>
  </head>
  <body>
    <h1>Media Queryを調べる</h1>
    <p>ウィンドウサイズを変更した後にボタンクリック</p>
    <form>
      <input type="button" value="Media Queryチェック">
    </form>
    <output></output>
  </body>
</html>
```

SAMPLE CODE JavaScriptのコード(sample.js)

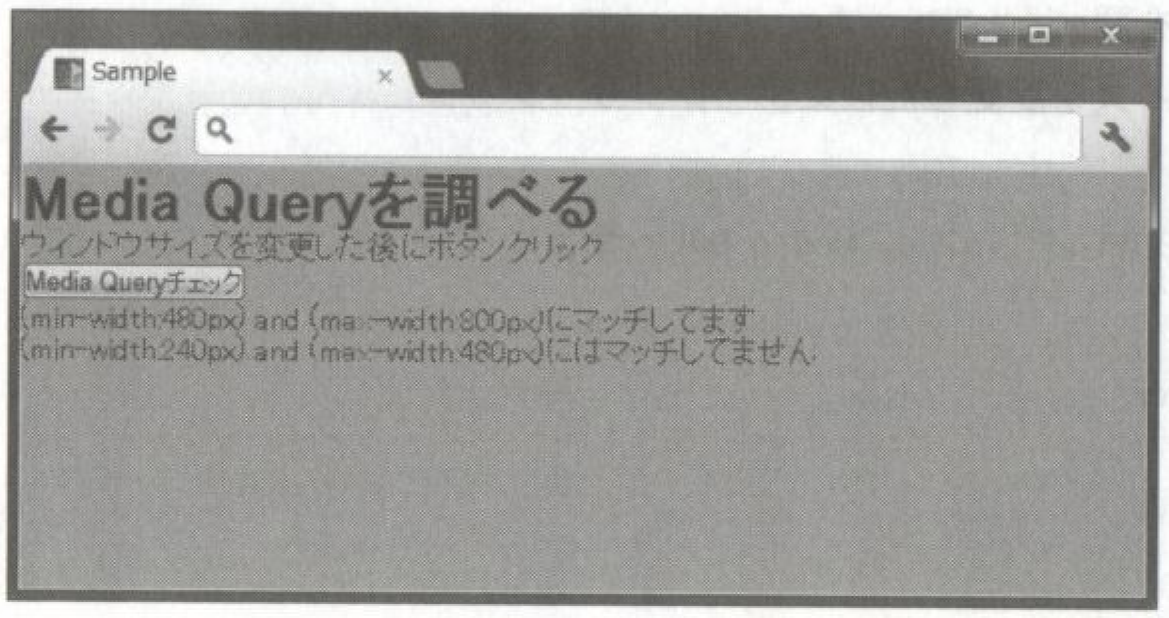
```
window.addEventListener("load", function(){
  var ele = document.getElementsByTagName("output")[0];
  // ボタンにclickイベントを割り当て
  var btn = document.getElementsByTagName("input")[0];
  btn.addEventListener("click", function(){
    // 調べるMedia Query文字列を指定。IE10はmsMatchMedia
    var flag = window.matchMedia("(min-width:480px) and (max-width:800px)").matches;
    if (flag === true) {
      ele.innerHTML = "(min-width:480px) and (max-width:800px)にマッチしてます<br>";
    }else{
      ele.innerHTML = "(min-width:480px) and (max-width:800px)にはマッチしてません<br>";
    }
    // 調べるMedia Query文字列を指定。IE10はmsMatchMedia
    flag = window.matchMedia("(min-width:240px) and (max-width:480px)").matches;
    if (flag === true) {
      ele.innerHTML += "(min-width:240px) and (max-width:480px)にマッチしてます";
    }else{
```

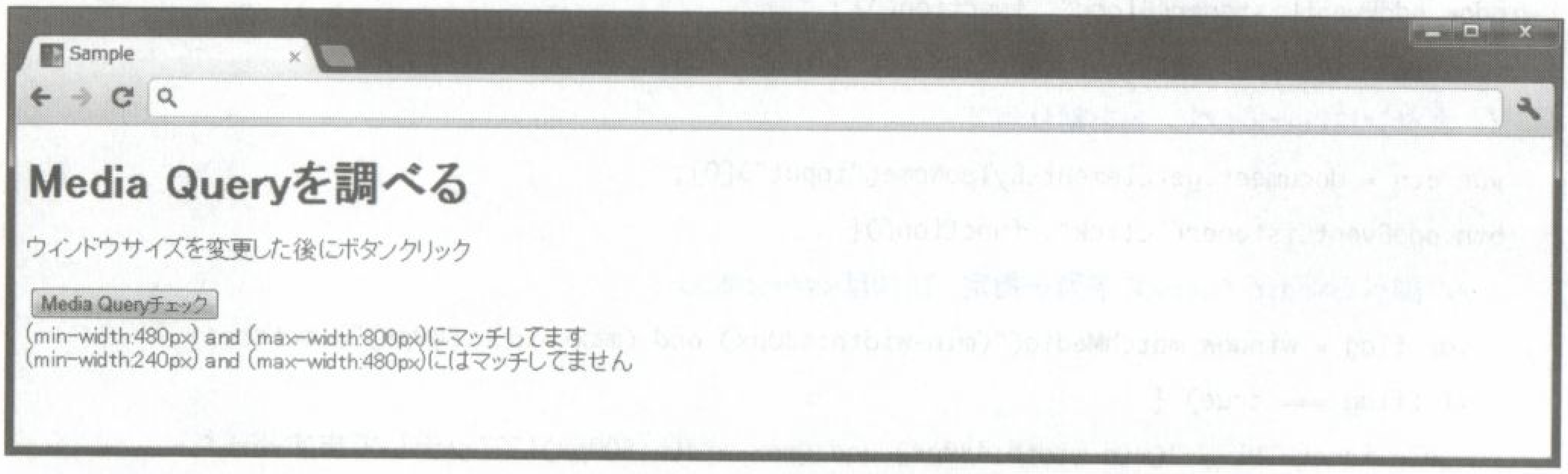
```
      ele.innerHTML += "(min-width:240px) and (max-width:480px)にはマッチしてません";
    }
  });
}, false);
```

ブラウザでindex.htmlを表示すると、ウィンドウの横幅に応じてページの背景色が変化するCSSファイルが読み込まれています。チェックボタンをクリックすると、対応状況が表示されます。



ウィンドウサイズを変化させた後で、再度チェックボタンをクリックすると、対応状況が表示されます。





ONEPOINT メディアクエリが変化したら処理するには「window.matchMedia()」メソッドと「addListener()」メソッドを使う

メディアクエリが変化したら処理するには、「window.matchMedia()」メソッドと「addListener()」メソッドを使います。パラメータには、マッチさせるクエリ文字列を指定します。また、メディアクエリが変化した場合に処理するため、「addListener()」メソッドを使います。「addListener()」のパラメータには、メディアクエリが変化した際に呼び出される関数を指定します。たとえば、メディアクエリが変化した場合にアラートダイアログを表示するイベントを設定するには、次のようにします。

```
var mm = window.matchMedia("(min-width:480px) and (max-width:800px)");
mm.addListener(function(){
  // Media Queryが変化した場合の処理
  alert("Media Queryが変わりました");
});
```

「addListener()」メソッドで設定したイベントを解除するには、「removeListener()」メソッドを使います。

なお、IE10で「matchMedia()」メソッドの使用するには、「msMatchMedia()」のように、ベンダープレフィックスを付ける必要があります。

関連項目 ▶▶▶

● CSSのメディアクエリが変化したら処理する p.438

INDEX

記号

A

B

C

D

E

F

J

K

L

M

T

U

か行

さ行

た行

な行

は行

ま行

や行

ら行

わ行

■著者紹介

古籏（ふるはた） 一浩（かずひろ）　49冊目。なるべく最新の情報を入れようとしましたが、HTML5は仕様変更が多い上にブラウザによって実装がまちまちで何とも大変。あと、HTML5は一人で把握できる規模を越えてるなあとも感じました。何にせよ面白いネタであることには間違いありません。

http://www.openspc2.org/

編集担当：吉成明久

JavaScript逆引きハンドブック

2012年9月10日　第1刷発行
2015年8月 4日　第3刷発行

著　者　古籏一浩
発行者　池田武人
発行所　株式会社　シーアンドアール研究所
　本　社　新潟県新潟市北区西名目所4083-6(〒950-3122)
　東京支社　東京都千代田区飯田橋2-12-10日高ビル3F(〒102-0072)
　電話　03-3288-8481　FAX　03-3239-7822
印刷所　株式会社　ルナテック

ISBN978-4-86354-108-5 C3055

Printed in Japan

PHP逆引きハンドブック

カバーデザイン:秋田 勘助(オフィス・エドモント)

ISBN978-4-86354-108-5

C3055 ¥3800E

9784863541085

C&R研究所

1923055038004

定価：本体3,800円＋税

◆本書で解説しているHTML5のAPIの一覧

- Geolocation API
- Video&Audio
- Web Storage
- Web Workers
- File API
- Canvas(2D Context)
- Canvas(WebGL)
- Offline Applications
- Touch Event
- Device Orientation API
- Navigation Timing/Performance Timing
- Cross Document Messaging
- Fullscreen API
- Vibration API
- Battery Status API
- WebRTC/MediaStream Processing API
- HTML Media Capture
- GamePad API
- WebSocket
- Indexed Database
- Web Audio API
- Pointer Lock API
- Page Visibility API
- Drag&Drop
- Web Notifications
- SVG
- Shadow DOM

CONTENTS

JavaScript逆引きハンドブック

HTML5/CSS3対応！

古籏一浩◆著

C&R研究所